Reasoning over Data: Analogy-based and Transfer Learning to improve Machine Learning

Doctoral thesis

Submitted and publicly presented on June 24, 2024

in partial fulfilment for the award of the degree of

Doctorat de l'Université de Lorraine
(mention informatique)

By

Esteban Marquer

IAEM – Lorraine doctoral school

Laboratoire Lorrain de Recherche en Informatique et ses Applications – UMR 7503

Acknowledgements

Many have assisted and supported me during the scientific journey that led to this manuscript, and would like to thank them all. One may notice that no names are given below. I didn't want to put any form of order between the people that assisted me, as all the help and support I received has been invaluable. Also, if I was to write the name of every person that helped, directly or indirectly, during my Ph.D, the first dozens of pages of this already long manuscript would have looked like the credits screen of a blockbuster film. For my supervisors, for the members of the jury, they can be found on the cover of this manuscript. For the others, I hope you can recognize yourselves, and I want you to know I am forever thankful to you.

In particular, I would like to thank my supervisors, who watched over me with benevolence. Their contagious motivation, their support, and the opportunities they made available to me drove me to give the best of myself, even in the hardest periods of the four years of my thesis work. And that's without even mentioning the invaluable guidance, scientific expertise and technical help they offered during the thesis. Once again, a big "thank you!" to them.

I am grateful to all the members of my jury, for their insightful feedback and constructive criticism, which significantly enhanced the quality of this work.

I also want to thank my colleagues and my collaborators, my interns and my students, for the work we did together and all the interesting discussions we had. I am particularly grateful to them, as well as to everybody I collaborated with, as they taught me many things as a person and as a scientist.

I am especially thankful my friends and family, who were understanding of the contraints of a Ph.D. student and supportive of my efforts. Without them, I probably wouldn't have finished this doctoral journey on such a happy note, if at all. Among my friends, I am indebted to all who helped in reviewing the manuscript in its early early stages. I know the size of this thesis can be intimidating, and every bit of help counted.

I extend my thanks to all my teachers and mentors, that before even starting this thesis, helped me forge my values and my curiosity in the fire of my admiration to them. They are models and inspirations, even if they might not realize how impactful their presence has been for me. You will forever remain in my heart.

Résumé

Ces dernières années ont été marquées par un regain d'intérêt pour le potentiel de l'inférence analogique et de la détection des analogies, avec des applications fructueuses dans l'apprentissage automatique pour la découverte et la génération d'images, de textes et de données structurées telles que les graphes de connaissances, ou encore la détection des relations entre et au sein d'images, de textes, ou de données structurées. Si certains de ces travaux reposent sur une compréhension intuitive de l'analogie, des efforts considérables ont été déployés depuis l'Antiquité pour définir les analogies de la manière la plus précise possible. Les analogies sont un élément clé de la cognition humaine et peuvent être considérées comme un mécanisme d'abstraction qui identifie les similitudes et les différences entre différentes situations, et comme un outil de raisonnement permettant d'adapter des solutions connues à de nouvelles situations. Lorsque l'on raisonne par analogie, l'objectif est d'adapter la solution d'un problème connu ou source, qui est suffisamment similaire au problème réel ou cible. Ce processus implique un transfert entre le contexte du problème source (le problème et sa solution) et le contexte du problème cible. Au cours des 50 dernières années, différents aspects de la notion de proportions analogiques (PAs) ont été explorés. Une PA est généralement composée de quatre éléments, écrits $A : B :: C : D$ lorsque le rapport entre A et B est conforme à celui entre C et D. Cet outil formel a été décliné pour couvrir de nombreuses interprétations différentes de ce qu'une analogie peut être, avec n'importe quel nombre d'éléments. Dans notre travail, nous explorons différentes méthodes pour aborder la détection des PAs et la résolution d'équations analogiques dans différents domaines, en utilisant l'apprentissage profond. En particulier, nous étudions la morphologie des mots et la désambiguïsation du sens cible pour les mots en contexte, deux domaines pour lesquels notre modèle est plus performant que l'état de l'art, ainsi que la sémantique des cadres, pour laquelle nous obtenons des résultats encourageants. Nous étudions également CoAT, un système de raisonnement à partir de cas (CBR) basé sur le transfert analogique. Avec CoAT, nous obtenons des succès significatifs dans la mesure de la compétence des cas et dans la tâche de compression de la base de cas.

Abstract

Recent years have seen a renewed interest in the potential of analogy detection and analogical inference, with successful applications in Machine Learning (ML) to the retrieval and generation of images, of text, and structured data such as knowledge graphs, but also the detection of relations between and within images, texts, and structured data. While some of those works are based on an intuitive understanding of analogy, significant effort has been made since the antiquity to define analogies as accurately as possible. Analogies are a key component of human cognition, and can be viewed as an abstraction mechanism that identifies similarities and differences between different situations, and as a reasoning tool to adapt known solutions to new situation. When reasoning by analogy, the goal is to adapt the solution of a known or source problem, which is sufficiently similar to the actual or target problem. This process involves a transfer between the context of the source problem (the problem and its solution) and the context of the target problem. In the past 50 years, different aspects of the notion of Analogical Proportions (APs) have been explored. An AP is typically composed of four elements, written $A : B :: C : D$ when the ratio between A and B is conform with the one between C and D, and this formal tool has been declined to cover numerous different interpretations of what an analogy can be, with any number of elements. In our work, we explore different methods to tackle the detection of APs and solving of analogical equations on different domains, using Deep Learning (DL). In particular, we study word morphology and Target Sense Disambiguation for words in context, two domains for which our model outperform the State of the Art (SotA), as well as in frame semantics, where we obtain encouraging results. We also study CoAT, a Case-Based Reasoning (CBR) system based on analogical transfer. With CoAT, we encounter significant success in the measure of case competence and in the task of case base compression.

Contents

Chapter 1

Introduction

Analogies are a key component of human cognition, and can be viewed as an abstraction mechanism that identifies similarities and differences between different situations, and as a reasoning tool to adapt known solutions to new situations. When reasoning by analogy, the goal is to adapt the solution of a known or source problem, which is sufficiently similar to the actual or target problem. This process involves a transfer between the context of the source problem (the problem and its solution) and the context of the target problem. Recent years have seen a renewed interest in the potential of analogy detection and analogical inference, with successful applications in Machine Learning (ML) to the detection of relations, the retrieval and generation of images, of text, and of formal knowledge units such as knowledge graphs. While some of those works are based on an intuitive understanding of analogy, significant effort has been made since the antiquity to define analogies as accurately as possible. In particular, a notion that garnered a lot of interest in the past 50 years is the one of Analogical Proportion (AP). An AP is typically composed of four elements A, B, C, D, and signifies that the relation from A to B is analogous to the one between C and D, written $A : B :: C : D$. This formal tool has been declined to cover numerous different interpretations of what an analogy can be, with any number of elements.

The interest of morphology as a benchmark to study APs. Applying analogical reasoning is not as easy as it seems. Word morphology is a very interesting benchmark for the study of APs, as it is a form character string analogy, and can usually be generalized to analogy between strings of symbols. Morphological data is accessible for many languages at the era of internet, and the transformations that can happen by the mechanisms of morphological inflexion have been extensively studied from a linguistic point of view. Additionally, while morphological inflexion is very regular for the most part (consider the AP "*dog*" : "*dogs*" :: "*cat*" : "*cats*" adding the suffix $-s$ to *cat* and *dog*), small variations and irregularities are present that can be hard to predict without linguistic knowledge (consider the AP "*dog*" : "*dogs*" :: "*bus*" : "*buses*" where the suffix $-s$ becomes $-es$ with *bus*). In such a setting, analogical reasoning can be used to leverage examples of morphological transformations and perform explainable predictions with minimal effort. For instance, in the 1980s a tool named Copycat [HM95] was proposed to solve APs between strings of characters, allowing, for instance, to find the solution $x =$ "*pqs*" to the AP equation "*abc*" : "*abd*" :: "*pqr*" : x, or $x =$ "*cats*" for the equation "*dog*" : "*dogs*" :: "*cat*" : x. Other symbolic approaches have since been developed to tackle APs in the domain of morphology, and while their performance gets closer to human behavior, the irregular aspect of word morphology is a common limitation.

The ANN framework to tackle morphological APs using Deep Learning (DL). In that context, we adapt and extend upon a recent DL approach based on APs [LPR19] that outperform more rigid formulations on the domain of word semantics. We propose the Analogy Neural Network framework (ANN framework), which includes the Analogy Neural Network for classification (ANNc) and the Analogy Neural Network for retrieval/generation (ANNr), two lightweight DL models to tackle analogy detection and solving, as well as a data augmentation process. The framework is formally grounded in the works on APs, and benefits from the flexibility of DL models to deal with irregularities in the data. To explore the performance of our approach, we developed the Siganalogies dataset covering more than 80 languages. Our approach outperforms symbolic approaches by a significant margin on analogy detection, analogy solving by retrieval, and analogy solving by generation, in some cases by more than 75% of accuracy. We perform extensive ablation experiments which allow us to refine the ANN framework. Having large amounts of data in many languages allows us to explore morphological APs across languages, which reveals interesting similarities between our model and the diachronic relatedness of languages[1].

Extending the success of the ANN framework to semantics and Pretrained Language Model (PLM). Motivated by the successes of the ANN framework on morphological APs, we adapt the framework to application domains beyond word morphology. For instance, we develop a model based on Bidirectional Encoder Representations from Transformer (BERT) and the ANN framework to solve Target Sense Verification (TSV), and outperform the State of the Art (SotA) on the Words-in-Context-TSV (WiC-TSV) benchmark. We also propose a system leveraging semantic APs based on frame semantics to tackle Frame Semantic Role Labeling (FSRL), on the FrameNet-1.7 (FN1.7) dataset. This new method appears flexible and while some challenges remain in the selection of the source for the analogical transfer of semantic role labels, it has the potential to outperform SotA methods.

Analogical Proportions beyond the ANN framework. Finally, we explore the potential of a dataset Complexity Measure for Analogical Transfer (CoAT) for several aspects of case base maintenance, an important problem in the domain of Case-Based Reasoning (CBR). CBR approaches consider a set of cases, that describe situations and their outcomes, or problems and their solutions, and rely on this case base to solve new problems or predict the outcome of new situations. In that context, we were able to define a measure of the competence, or usefulness, of cases in the case base, which allowed us to improve the quality of case bases to solve synthetic and real-wold classification tasks.

Structure of the document. This thesis is separated in 3 parts. First, Part I contains introductions to the key notions and approaches mentioned in this thesis, namely analogy, DL, and morphology. Then, Part II describes our contributions on the ANN framework in morphology, while Part III regroups our contributions to TSV, frame semantics, and CBR.

Main contributions. Our contributions to the literature discussed in this thesis are as follows:

- [Als+21a]: a first conference article published on the ANNc and CNN-based word embedding (CNN-emb) models for analogy detection, mainly mentioned in Chapters 6 and 8 and Section 7.3;
- [Als+21b]: a workshop article in which we describe experiments on training ANNc with multilingual data and transferring monolingual ANNc to other languages, as described in Section 8.1;
- [Als+21c]: an unpublished document listing some of our preliminary experiments on ANNc (see Subsections 6.3.3 and 7.3.3) and on analogy solving with ANNr (see Subsections 6.4.2, 6.4.3, 7.4.2 and 7.4.4), as well as some ablation studies on the two models (see Section 7.5);
- [Mar+22a]: a second conference article, in which we describe a more elaborate retrieval approach using ANNr for analogy solving, reported in Subsection 6.4.2 and Section 7.4;
- [Cha+22]: a workshop article in which we present an alternative to the CNN-emb model, namely the AE-based word embedding (AE-emb) model, to solve some of its limitations for analogy solving, mentioned in Subsections 6.1.2 and 7.4.3;
- [MMC22]: a workshop article in which we perform several experiments using transfer learning without adaptation, allowing us to study the impact of the postulates of AP (see Section 7.6), to explore the limits of the generalization of ANNc, and to extend the cross-lingual transfer experiments from [Als+21b] (see Section 8.2);
- [MC24]: a special issue article that summarizes our main contributions with the ANN framework and is indirectly referenced all over Part II, it also extends our analogy solving results, by combining the AE-emb and ANNr, as well as with other complementary experiments;

[1]The diachronic relatedness of languages compares their evolution through history.

- [Mar+22b]: the Siganalogies dataset we developed for all our experiments on APs in morphology, built upon Sigmorphon 2016 Task 1 (Sig16), Sigmorphon 2019 Task 1 (Sig19), and Japanese Bigger Analogy Test Set (JBATS), and detailed in Chapter 5;
- [Als+22a]: a workshop article in which ANNc is applied on medical records, a collaboration that is mentioned in Chapter 9 and extended by [Als+22b];
- [Zer+22]: a third conference article, which describes how we use ANNc for TSV in WiC-TSV, reported as the Chapter 10;
- [Mar+23]: a workshop article that presents our approach to case and case base competence based on CoAT, containing a number of experiments on synthetic data, detailed and completed in Chapter 12;
- [Bad+23]: a workshop article proposing perspectives on the use of CoAT in the CBR domain, presenting several directions of work and mentioned in Chapter 12.

Another article describing the contributions to FSRL in FN1.7 (see Chapter 11) is under review at the time of writing.

Other related scientific activities. In addition to these publications, and within the frame of this thesis, I co-organized the *Analogies: from Theory to Applications* (ATA) workshop in the *International Conference on Case-Based Reasoning* (ICCBR) in 2022 and 2023, and am a co-editor of the Springer Special Issue *Annals of Mathematics and Artificial Intelligence: Mathematical Foundations of Analogical Reasoning and Applications.*

In parallel to other reviewing activities [Ara+21, among others], I am a member of the program committee of the workshop on *Interactions between Analogical Reasoning and Machine Learning* (IARML) from 2022 to 2024 (at the time of writing), that was hosted at the *International Joint Conference on Artificial Intelligence and European Conference on Artificial Intelligence* (IJCAI-ECAI) in 2022, and at the *International Joint Conference on Artificial Intelligence* (IJCAI) in 2023 and 2024.

I also supervised Master students and interns, with noteworthy collaborations [Als+21a; Als+21b; Als+21c; Cha+22; Mar+22a].

The work reported in this thesis lies within the scope of the Inria Project Lab *Hybrid Approaches for Interpretable Artificial Intelligence* (HyAIAI) and *Analogies: From Theory to Tools and Applications* (ANR project AT2TA, grant ANR-22-CE23-0023) projects. In particular, I had the opportunity to actively participate in the AT2TA project, that strongly aligns with my thesis topic.

Part I
Preliminaries and state of the art

In this first part, we introduce the key concepts used in this thesis. Firstly, in Chapter 2 we introduce the subtleties of the notion of analogy and describe the more precise notion of APs. We describe several approaches to formalize the properties of APs. Secondly, the basic concepts related to DL, that are necessary to understand this thesis, are introduced in Chapter 3. Finally, the domain of application for the approaches and experiments we present in Part II is morphology. We explain in Chapter 4 what morphology is, including a description of some key approaches.

Chapter 2

Analogy and analogical proportions

Chapter contents

Analogy can describe distinct but related notions depending on the context. We attempt to give an overview of these notions and how they are related in Section 2.1. Among these notions, we focus in particular on the one of Analogical Proportion (AP). As defined in Subsection 2.1.1, an AP is a quaternary relation between elements usually of the same nature, written $A : B :: C : D$ and read "A is to B as C is to D". To illustrate the versatility of APs, we list some interesting applications in Section 2.2.

In Section 2.3 we introduce the formal framework we use for our approaches, which follows the seminal work of Lepage and Ando [LA96] by exploiting the postulates of APs. We introduce some limitations of the formal framework in Subsection 2.3.2, which will be further discussed later in Subsection 6.3.2, Section 7.6, and Appendix D. To give the reader an overview of possible formulations of the problem of analogy, we also provide a brief introduction to other key formal frameworks in Section 2.4.

2.1 The vast notion of analogy

Analogy has been used in a variety of settings and refers to similar but distinct notions. For instance, it may refer to the figure of speech related with metaphor, to analogies between four elements (that we will describe as APs, see also Subsection 2.1.1), or as the Cambridge dictionary defines, to "a comparison between things that have similar features, often used to help explain a principle or idea".

While there is not general consensus on the different notions of analogies, it is generally accepted that they can be reformulated as APs. For instance, Barbot, Miclet, and Prade [BMP19, Section

2.1] separate analogies into four notions and provide a categorization of analogies in language, that we summarize hereafter. Among them, *relational proportions* (Subsection 2.1.2) follow the same writing as APs (Subsection 2.1.1), while *simile* (Subsection 2.1.3) and *metaphor* (Subsection 2.1.4) can be seen as APs where some elements are not explicitly expressed.

For a more historical analysis of the notion of analogy, we recommend the first chapter of [Lep03, in French]. Among the notions accounted for in the aforementioned chapter, the distinction between continuous (analogical) proportions and discontinuous (analogical) proportions, made by Aristotle, is described in Subsection 2.1.5.

2.1.1 Analogical Proportions

Barbot, Miclet, and Prade [BMP19] define APs as relations between four elements of the same nature. They follow formulations like "A is to B as C is to D", usually formalized as $A : B :: C : D$, as in Example 2.1.

The relational view of APs is commonly used, and states that $R(A, B) = R(C, D)$, with the relations $R(A, B)$ and $R(C, D)$ corresponding to the ratios $A : B$ and $C : D$, while the equality corresponds to the conformity of ratios usually noted $::$. Another commonly used view of APs is the functional view, that is based on a function (or transformation) f and reformulates APs as $A : f(A) :: C : f(C)$. This can be seen as a special case of relational AP, when we consider the relation between an object A and its image $f(A)$. APs express both similarities and dissimilarities [PR14], and follow strong properties that we detail in Section 2.3.

Example 2.1: A first AP

Let us consider the AP:

"a cat is to a kitten as a dog is to a puppy"

$$\text{“}\textit{cat}\text{”} : \text{“}\textit{kitten}\text{”} :: \text{“}\textit{dog}\text{”} : \text{“}\textit{puppy}\text{”}.$$

Multiple similarities and differences are expressed in this example:

- a cat and a kitten differ in the same way as a dog and a puppy, as a cat is the adult form of a kitten, and a dog is the adult form of a puppy;
- a cat and a dog are similar on the same aspects as a kitten and a puppy are, as a cat and a dog are adults while a kitten and a puppy are juveniles;
- a cat and a kitten are similar on features on which a dog and a puppy are similar, as a cat and a kitten are cats, while dog and puppy are different ways to designate dogs;
- on the same features, a cat and a dog differ in the same way as a kitten and a puppy.

2.1.2 Relational proportions

In the classification of Barbot, Miclet, and Prade [BMP19], relational proportions are very close to APs, with patterns of the form "A is to a as C si to c" or "A is the C of a", *e.g.*, "the eyes are the windows of the soul" or "the eyes are to the soul as windows are to a house". The main differences between APs and relational proportions are that the elements are not of the same nature anymore (a, c are of one kind while A, C are of an other) and c is usually obvious enough from the context of C to be omitted, resulting in the formulation "A is the C of a".

However, as done in the work of Barbot, Miclet, and Prade [BMP19], APs in language can be converted to relational proportions, and conversely.

2.1.3 Simile

Simile are figures of speech that transfer a property expected for an object to another object, written for instance "A is as a as C". An example of simile from Barbot, Miclet, and Prade [BMP19] is "I am as hungry as a wolf", in which the hunger that is expected from a wolf is transferred to "I".

Simile can be reformulated into forms similar to APs, for instance "A is as a as C" would become "a is to A as a is to C", Barbot, Miclet, and Prade considers them as special cases of relational proportions. However, the fact that A and C are different but the same element is used to compare to both of them may cause issues with some properties detailed in Section 2.3. To be more precise, the Exchange of the Means postulate states that from "A is to B as C is to D" we also have "A is to C as B is to D". With "a is to A as a is to C" however, we would have that "a is to a as A is to C", which implies that A and C equal, as a and a are. We discuss this issue in more detail in Subsection 2.3.2.

2.1.4 Metaphors

Metaphors describe an object using a different object, with formulations such as "A is like C", *e.g.*, "My teacher is a dragon". Compared to, for instance, relational proportions, metaphors do not express either a nor c, and are "more allusive than simile" [BMP19]. Indeed, it is significantly harder to determine with regards to what "my teacher is a dragon" without any point of reference, than if we formulate it as the simile "my teacher is like a dragon: frightening when angry". Metaphors, given enough information, can be reformulated as simile and therefore as APs, with the same constraints as simile.

2.1.5 Continuous proportions and discontinuous proportions

A specific case of AP is when B and C the same object: "A is to B as B is to D". This distinction was already made by Aristotle according to [Lep03, page 41], and does not cause the same kind of issues as simile with regards to the postulates of APs (see Subsection 2.1.3). Continuous APs were for instance used by Reed, Zhang, et al. [Ree+15] to produce sequences of images for animations, by answering the question "what is to the current image as the previous image was to the current one (that is, the next frame in the animation)", leveraging a model designed for discontinuous APs. The relation between continuous proportions and Boolean models of analogies is extensively discussed by Leemhuis and Özçep [LÖ23].

2.2 Applications of Analogical Proportions

APs can be used to formalize different types of tasks, in particular, analogy detection and analogy solving, that can take several variants including the analogical inference principle. We detail these tasks in Subsection 2.2.1.

While it is not possible to completely survey existing approaches using APs, we provide a first overview in Subsection 2.2.2. In later chapters, we detail approaches using APs with DL (Section 3.4) and applications to word morphology (Subsection 4.3.2).

2.2.1 Analogy detection, analogy solving, and analogical inference

Two main tasks are associated with APs: analogy detection and analogy solving. The analogical inference principle was used for instance in [BL22; Bay+07; Cou+17a; Hug+19; MBD08], and combines analogy detection and analogy solving. All three tasks are discussed in this subsection.

Analogy detection. Analogy detection consists in identifying whether a quadruple A, B, C, D forms a valid AP $A : B :: C : D$ or not. This task is of prime importance, as analogy detection relates to the ability to properly define what an AP is, however in some situations the boundary between valid and invalid AP is not clearly defined.

In cases like the arithmetic and geometric proportions (see Equations (2.3) and (2.4)), the notion underlying AP is well defined, and analogy detection is a simple process.

However, if we consider morphological APs, that deal with the structure of words as we will define in Chapter 4, we can have cases such as:

$$\text{“}\textit{cat}\text{”} : \text{“}\textit{cats}\text{”} :: \text{“}\textit{dog}\text{”} : \text{“}\textit{dogs}\text{”}$$
$$\text{“}\textit{cat}\text{”} : \text{“}\textit{cats}\text{”} :: \text{“}\textit{bus}\text{”} : \text{“}\textit{buses}\text{”}$$
$$\text{“}\textit{cat}\text{”} : \text{“}\textit{cats}\text{”} :: \text{“}\textit{child}\text{”} : \text{“}\textit{children}\text{”}$$
$$\text{“}\textit{cat}\text{”} : \text{“}\textit{cats}\text{”} :: \text{“}\textit{sheep}\text{”} : \text{“}\textit{sheep}\text{”}.$$

While it is easy to agree that the first AP is valid in terms of morphology (we add "-s" on both sides), it can be argued that the further down we go the less acceptable the AP is, as more and more non-basic morphological information is needed: for the second AP we need to adapt the suffix "-s" into "-es" based on morphological rules, for the third one we need to change the suffix completely, and for the last one we need to know that "sheep" is invariant. Determining where to draw the line is therefore one of the main challenges of analogy detection. In ML terms, analogy detection is usually seen as a binary classification task, where a model has to label a given quadruple as a valid or an invalid AP.

Considering that in some case analogies have infinitely many solutions [HM95], instead of a strict valid/invalid decision some approaches determine the "degree of analogicality" [LG14; MBD08; MYZ13; Mur22; Mur+20] of the quadruple. It is then possible to position the quadruple on the valid to invalid AP spectrum, or, using a more cognitive view of APs [BMP19; Mit21], on the spectrum of very relevant to less relevant analogies.

To summarize, analogy detection involves the ability to determine how to compare the objects, *i.e.*, which common aspects are used for the analogical relation. For instance, for arithmetic proportions (Example 2.2 and Equation (2.4)) we can compare the arithmetic mean of the means and the extremes, and for our work on morphological APs (see Chapter 4 and Part II) we compare the morphological features of words. Analogy detection also requires the ability to determine how "analogical" a quadruple is, either in a binary manner or in a continuous manner.

Example 2.2: Example of arithmetic proportion

An arithmetic proportion is a type of AP $A : B :: C : D$ between numbers, where the ratio (:) is the difference ($-$), and the conformity of ratios (::) is the equality ($=$). As can be seen in the example:

$$8 : 2 :: 12 : 6,$$
$$8 - 2 = 12 - 6 = 6,$$

an arithmetic proportion compares difference between the first (8) and the second (2) elements with the difference between the third (12) and fourth (6) elements. It is also possible to compare the arithmetic mean of the means, *i.e.*, the second (2) and third (12) elements, with the extremes, *i.e.*, the first (8) and fourth (6) elements:

$$\frac{8+6}{2} = \frac{2+12}{2} = 7.$$

Analogy solving. Analogy solving is the process of completing an incomplete AP, in particular an analogical equation in which one of the elements is unknown, that we write $A : B :: C : x$ where x is unknown.

Analogy solving raises two important questions, that we formulate as Solvability and Uniqueness in Table 2.1, with examples in Example 2.3:

- is it always possible to solve an analogical equation (Solvability)? If not, how do we determine if a triplet A, B, C forms a solvable analogical equation $A : B :: C : x$ [Mur22]?
- if an AP is solvable, is there only one solution (Uniqueness) [DL23]?

Example 2.3: Analogical equations with more or less than one solution

Let us consider APs between strings of symbols, where the allowed transformation include insertion and deletion of symbols, and ε represents an empty string.

If we take the following analogical equation:

$$\varepsilon : \text{“}a\text{”} :: \text{“}bb\text{”} : x,$$

it can reasonably accept the following solutions: $x = \text{“}abb\text{”}$, $x = \text{“}bab\text{”}$, $x = \text{“}bba\text{”}$.

> If we take the analogical equation:
>
> $$\text{“}a\text{”} : \varepsilon :: \text{“}bb\text{”} : x,$$
>
> with the ratio “a” : ε corresponding to removing an a from the left element, it is reasonable to consider that there is no solution, as there is no a to remove in bb.

To our knowledge, most of the effort on modelling APs focuses on analogy solving [LG14; MBD08; Mur22; Mur+20; Ree+15; SZF15, among many other]. In ML terms, analogy solving is usually seen as a retrieval [LG14; LPR21; MBD08; SZF15] or generation task [LYZ09; Mur+20; Ree+15].

The basic formulation $A : B :: C : x$ (x unknown) of analogy solving can be generalized to finding all missing information from some A, B, C, D such that $A : B :: C : D$ holds [BL22]. When only D has missing information, this generalization corresponds to the analogical inference principle described hereafter.

Analogical inference and analogy preservation. A specific case of analogy solving leverages the analogical inference principle, attributed to Pirrelli and Yvon [PY99] by Prade and Richard [PR21], that is defined as follows, for elements of the same nature with features X partitioned in two subsets X_1, X_2:

$$\frac{\forall i \in X_1, \quad A_i : B_i :: C_i : D_i \quad \text{holds}}{\forall j \in X_2, \quad A_j : B_j :: C_j : D_j \quad \text{holds}}, \tag{2.1}$$

in other words, if for some of their features (the ones in X_1), A, B, C, D form an AP, then it is also the case for the remaining features (X_2).

This principle has been used for instance by Couceiro, Hug, et al. [Cou+17a], Badra and Lesot [BL22], Lieber, Nauer, and Prade [LNP21], Lepage and Denoual [LD05], and Couceiro and Lehtonen [CL24]. It combines analogy detection with analogy solving to find the missing features F' of an element D for which we know features X_1. To do so, we first find three elements A, B, C for which $\forall i \in X_1, \ \ A_i : B_i :: C_i : D_i$ holds with an analogy detection method, and for which we know the values for features X_2. Then we apply an analogy solving technique to solve $\forall j \in X_2, \quad A_j : B_j :: C_j : x$, and use the solutions to complete D.

In the work of Couceiro, Hug, et al. [Cou+17a], functions that maintain this principle are called analogy preserving functions. Formally, a function $f : X_1 \mapsto X_2$ is analogy preserving if the following holds for all $A, B, C, D \in X_1$:

$$\frac{A : B :: C : D \quad \text{holds}}{f(A) : f(B) :: f(C) : f(D) \quad \text{holds}}. \tag{2.2}$$

Relation identification. One specific task related to analogy detection and analogy solving is the identification of the relation (or the function f in the functional reading) underlying an AP [GDM16; Pey+19]. This process can be used to explain a relation between two elements, but also to solve an AP by first identifying the relation between the first two elements then apply it on the third one to find the fourth missing element. This is for instance applied in [MDC17; Mur+20, see Subsection 2.4.2].

2.2.2 Approaches using Analogical Proportions

The principles of analogy detection, analogy solving, and analogical inference have been used in a wide variety of settings, that are identifiable by the nature of the manipulated data, the nature of the underlying relations, and the end goal.

There have been approaches that manipulate, among other data types:

- images, whether on the situation depicted [Bay+07; Bit+23; LTC17; Ree+15; SZF15] or the pixel themselves [Lep14];
- knowledge graph entities [JCM23];
- undirected graphs [Ant23] and trees [Bod09; ZFL22];

- Boolean data [BMD07; Cou+17a; Cou+18];
- tabular data [proposal by MC22];
- strings of symbols [HM95; MDC17];
- text, at different levels of granularity: strings of characters [DL23; FL18; MC24; MYZ13; Mur+20], words [DGM16; GDM16; LPR21; Mik+13], sentences [Afa+21; Afa+22; LD05; ML23; TWL20; WL20; ZM20], or entire documents [Als+22a; Als+22b];
- feature vectors in vector spaces [DZF19; DGM16; LG14; LPR21; MBD08; Mik+13; RA73; ZM20], including embedding spaces (see Section 3.3) and binary or nominal features.

Focusing on approches on textual data, analogy has been used for a wide variety of purposes:

- providing cognitively sound models of analogy, for instance [HM95; MDC17], and to the best of our knowledge, most of the literature on AP;
- evaluating the quality of the representation for embedding spaces [DGM16; Dum+88; GDM16; Mik+13];
- improving the quality of the representation for embedding spaces [DZF19; Kar+18];
- performing machine translation [LD05; TWL20]
- completing missing information [MC22], for instance for medical records [Als+22a; Als+22b];
- making parallels between scientific domains [SS22];
- transferring annotations (as we do in Chapter 11);
- performing or analyzing morphological inflexion [DL23; FL18; MC24; Mur+20, see also Subsection 4.2.1];
- exploring the limits of the reasoning of large language models reasoning [Ush+21; Yas+24; Zer+22].

As one can see, there is a wide variety of applications and approaches, even though we limit our selection to works where the use of APs is easily identifiable. For instance, other interesting references can be found in [PR21]. For this reason, we will detail only some approaches that are directly relevant to the work presented in this thesis. In particular, after presenting fundamental notions of DL in Chapter 3, we will describe various applications of analogies in DL and in particular in vector spaces in Section 3.4. Then, after presenting word morphology in Chapter 4, we will discuss applications of APs to word morphology in Subsection 4.3.2.

2.3 Axiomatic Analogical Proportions

In most of our work, we follow the setting defined by Lepage [Lep01; Lep03], which follows the seminal work of Lepage and Ando [LA96]. This axiomatic setting defines a set of postulates for analogical quadruples, and defines the notion of AP from these postulates. We detail these postulates in Subsection 2.3.1, and summarize them in Table 2.1, based on a categorization proposed in Subsection 2.3.3. Some of the postulates introduced are not suitable to some application contexts [Als+22a; Ant22], as discussed in Subsection 2.3.2. Further postulates are mentioned in Appendix D, but do not correspond the usual setting of APs.

We try to match the terminology from Lepage [Lep01] whenever possible for the postulates names, and mention other common terminologies that we are aware of [Afa+22; Ant22; LPR21; PR21].

2.3.1 The postulates of Analogical Proportions

The axiomatic setting from [LA96] is related to the common view of APs as geometric (Equation (2.3)) or arithmetic proportions (Equation (2.4)) or, in geometric terms, as parallelograms in a vector space (or parallelogram rule, Equation (2.5)):

$$\frac{A}{B} = \frac{C}{D}, \tag{2.3}$$

$$A - B = C - D, \tag{2.4}$$

$$\vec{A} - \vec{B} = \vec{C} - \vec{D}. \tag{2.5}$$

The various postulates proposed by Lepage were built, among other inspirations, on these proportions. In particular, the postulates described below correspond to properties verified by the geometric and arithmetic proportions and the parallelogram rule.

The setting defined in [Lep01] consists in two postulates:

- **Symmetry of Conformity**: if $A : B :: C : D$, then $C : D :: A : B$;
- **Exchange of the Means**: if $A : B :: C : D$, then $A : C :: B : D$.

In the work of Prade and Richard [PR21], Afantenos, Lim, et al. [Afa+22], Antić [Ant22], and Lim, Prade, and Richard [LPR21], Exchange of the Means is also called *central permutation*, while Symmetry of Conformity is called *symmetry*.

These two properties can be combined to obtain a total of 8 permutations of the AP $A : B :: C : D$ that also hold true:

- $A : B :: C : D$, the base form;
- $A : C :: B : D$, Exchange of the Means;
- $B : A :: D : C$, **Inversion of Ratio**, *inside pair reversing* [Als+21a], or *internal reversal*;
- $B : D :: A : C$, obtainable by Exchange of the Means followed by Symmetry of Conformity;
- $C : A :: B : D$, obtainable by Symmetry of Conformity followed by Exchange of the Means;
- $C : D :: A : B$, Symmetry of Conformity;
- $D : B :: C : A$, **Exchange of the Extremes**, or *extreme permutation*;
- $D : C :: B : A$, **Symmetry of Reading**, or *complete reversal.*

As stated by Lepage [Lep03, Section 4.1.2], Symmetry of Conformity can be replaced by Inversion of Ratio, and Exchange of the Means by Exchange of the Extremes, for the same 8 equivalent forms.

Additionally, Lepage [Lep03] proposed three other postulates:

- **Reflexivity of Conformity**: $A : B :: A : B$ is always true;
- **Identity**: $A : A :: B : B$ is always true;
- **Strong Identity**, or *strong inner reflexivity*: if $A : A :: B : C$, then $C = B$;
- **Strong Reflexivity of Conformity**, or *strong reflexivity*: if $A : B :: A : C$, then $C = B$.

Uniqueness is a more general form of Strong Identity and Strong Reflexivity of Conformity, and states that $A : B :: C : D \wedge A : B :: C : D' \implies D' = D$. In other words, a given triplet of elements A, B, C correspond to at most one forth element D. Uniqueness makes sense, for instance, for arithmetic and geometric proportions, where the solution of $d = c + a - b$ is unique, and there is at most one solution for $d = c \times \frac{a}{b}$ (see Equations (2.3) and (2.4)). Strong Identity and Strong Reflexivity of Conformity can be seen as applications of Uniqueness to Identity and Reflexivity of Conformity, respectively.

There is a postulate introduced by Lepage [Lep03, page 122] that we do not use in our work, namely, the Distribution postulate. It states that if we have an AP $A : B :: C : D$, and we are able to distinguish the features of the elements, then any feature of A can be found in B or C, and the same is true for D in place of A. If we write X the set of features and $X(A)$ the features expressed in A, Distribution can be expressed as $A : B :: C : D \implies X(A) \subseteq X(B) \cup X(C)$. The assumption that we are able to distinguish the features of the elements might appear trivial, in particular for symbolic approaches, but is unsuitable for continuous domain APs or APs where the features are not explicitly expressed.

2.3.2 Discussions and limitations of the axiomatic setting

While the postulates of APs seem reasonable when manipulating words, which is what they were designed for in the work of Lepage [Lep03], they can be criticized in other application domains [Als+22a; Ant22].

The limitations of Exchange of the Means. Let us consider the simile $A : B :: A : C$, with A, B, C distinct objects. By Exchange of the Means this simile would produce $A : A :: B : C$. The latter is counter intuitive, as we already discuss in Subsection 2.1.3: humans tend to prefer the simplest solutions [CV03], and considering that $R(A, A)$ is the equality, however $B \neq C$ which is not conform with $A = A$. This consideration led us, in Section 7.6 [see also MMC22], to explore some limitations of the axiomatic setting for training models of morphological APs, and in particular how accepting or refusing Exchange of the Means impacts the performance of our analogy detection models.

Other recent works reach similar conclusions, for example, Antić [Ant22, proof of Theorem 28] demonstrates that the formalisation of APs he proposes does not satisfy Exchange of the Means in general. The approach from Murena, Al-Ghossein, et al. [Mur+20] does not fit Exchange of the Means either. As described in Subsection 2.4.2, the approach of Murena, Al-Ghossein, et al. considers the complexity of the ratio $A : B$ in an AP $A : B :: C : D$, which may differ from the complexity of the ratio $A : C$ in the AP $A : C :: B : D$ after Exchange of the Means. This allows their approach to handle certain forms of simile, while still refusing $A : A :: B : C$ with A, B, C distinct objects.

Another situation in which Exchange of the Means is not suitable is when elements are spread in different domains. For instance, in the work of Sultan and Shahaf [SS22], $A : B$ is a ratio in one conceptual domain while $C : D$ belong to another one, as in "Earth is to the Moon as a nucleus is to an electron", and it might not be possible or convenient to define a ratio between domains. APs that include elements of different nature may exhibit the same kind of issue, for instance with animals and their class: "mammal is to dog as reptile is to crocodile" causes no issue, while when considering Exchange of the Means, "mammal is to dog as mammal is to cat" will create a similar issue as what we had with simile.

The interest of a Symmetry of Ratio postulate. In our work on medical records [Als+22a] for instance, symmetric relations, including the equality of elements, are considered among the underlying relations of the APs. As such, $A = B :: C = D \implies B = A :: C = D$, as equality is symmetric. To account for these situations, we consider the Symmetry of Ratio postulate, that states that if $A : B :: C : D$ holds, then $B : A :: C : D$ holds and $A : B :: D : C$ holds. In most cases application, Symmetry of Ratio does not hold. However, if it holds together with Symmetry of Conformity and Exchange of the Means (or any set of postulates resulting in the 8 equivalent forms), a direct consequence is that all 24 possible permutations of 4 elements are equivalent [Lep03, Theorem 2 page 119]. We discuss and detail different combinations of postulates, as well as their correspondence with models similar to the one of APs, in Appendix D.

Some challenges with Transitivity. In the work of Antić [Ant22], additional postulates are used, including multiple notions of Transitivity that expand upon the postulate that $A : B :: C : D \land C : D :: E : F \implies A : B :: E : F$. Taking an example from Lim, Prade, and Richard [LPR21], while

$$\text{“}\textit{nurse}\text{”} : \text{“}\textit{patient}\text{”} :: \text{“}\textit{mother}\text{”} : \text{“}\textit{baby}\text{”} \tag{2.6}$$

$$\text{“}\textit{mother}\text{”} : \text{“}\textit{baby}\text{”} :: \text{“}\textit{frog}\text{”} : \text{“}\textit{tadpole}\text{”} \tag{2.7}$$

are two semantically acceptable APs, the result of Transitivity is harder to accept:

$$\text{“}\textit{nurse}\text{”} : \text{“}\textit{patient}\text{”} :: \text{“}\textit{frog}\text{”} : \text{“}\textit{tadpole}\text{”}.$$

This difficulty is due to the coexistence of multiple distinct underlying relations used in the first two APs: the relation underlying the ratio $X : Y$ would be "X takes care of Y" in Equation (2.6), while in Equation (2.7) the underlying relation is closer to "X gives birth to Y".

2.3.3 Proposed categorization of the postulates

We propose to group the postulates into 3 categories, based on the type of property they represent:

- **permutation postulates:** postulates that state that if an AP $A : B :: C : D$ exists, then another AP exists, obtainable by permuting A, B, C, D;
- **existence postulates:** postulates that state that a particular AP always exists;
- **constraint postulates:** postulates that constrains the possible values of in an AP.

The postulates are summarized and separated in their respective groups in Table 2.1. We also include two postulates to completely cover the 8 equivalent forms of AP defined by Lepage [Lep01; Lep03], named respectively extra postulate 1 and extra postulate 2. Beyond making more explicit the kind of contraints enforced by each postulate, this categorization of postulate is helpful for the proposal we make in Appendix D to generalize the data augmentation process from Section 6.3.

Symmetry of Ratio is the only permutation postulate producing two distinct permutations: it states that it is possible to flip the ratio (:) on the left or on the right of the conformity of ratios (::). To be consistant with the other permutation postulates, we split it into Symmetry of Left Ratio and Symmetry of Right Ratio.

Note that in this categorization, we do not include Transitivity in any of its existing variants, among other reasons because it is not a widely accepted postulate.

Postulate	Symbol	Description
	Permutation postulates group	
Symmetry of Conformity	*SymC*	$A : B :: C : D \implies C : D :: A : B$
Exchange of the Means	*EM*	$A : B :: C : D \implies A : C :: B : D$
Inversion of Ratio	*IR*	$A : B :: C : D \implies B : A :: D : C$
Exchange of the Extremes	*EE*	$A : B :: C : D \implies D : B :: C : A$
Symmetry of Reading	*Rev*	$A : B :: C : D \implies D : C :: B : A$
Extra postulate 1	*Ex*1	$A : B :: C : D \implies C : A :: D : B$
Extra postulate 2	*Ex*2	$A : B :: C : D \implies B : D :: A : C$
*Symmetry of Ratio	*Sym* :: *Sym*	
**Symmetry of Left Ratio*	*Sym* ::	$A : B :: C : D \implies B : A :: C : D$
**Symmetry of Right Ratio*	:: *Sym*	$A : B :: C : D \implies A : B :: D : C$
	Existence postulates group	
Reflexivity of Conformity	*Ref*	$A : B :: A : B$
Identity	*Id*	$A : A :: B : B$
Solvability	*Solv*	$\forall A, B, C\ \exists D$ such that $A : B :: C : D$ holds
	Constraint postulates group	
Uniqueness	*Uniq*	$A : B :: C : D \wedge A : B :: C : D' \implies D = D'$
Strong Reflexivity of Conformity	*UniqRef*	$A : B :: A : D \implies B = D$, can be seen as Uniqueness restricted to Reflexivity of Conformity
Strong Identity	*UniqId*	$A : A :: C : D \implies C = D$, can be seen as Uniqueness restricted to Identity
Distribution	*Dist*	$A : B :: C : D \implies \mathcal{X}(A) \subseteq \mathcal{X}(B) \cup \mathcal{X}(C)$, with $\mathcal{X}(A), \mathcal{X}(B), \mathcal{X}(C)$ the features of A, B, C

Table 2.1: Known postulates used in axiomatic analogies, excluding transitivity. Asterisks indicate permutation postulates that are not accepted for APs.

2.4 Other formalizations of Analogical Proportions

As mentioned in Subsection 2.1.1, two specific readings coexist with the general reading of AP, namely the relational and the functional reading. The notion of AP is related to many other formalisms, and different readings gave birth to different formalizations, some of which are described below. We describe the Boolean, Kolmogorov complexity, Structure Mapping Theory (SMT), and CBR formalisms in Subsections 2.4.1 to 2.4.4, respectively. Other formalisms have been proposed, including a Galois theory [CL24] and several algebraic approaches [Lep03; SY04; SY07, among others]. Additional explanations on some of the formalisms presented in this section are available in the review by Prade and Richard [PR21].

Among others approaches, Prade and Richard [PR21] relate the DL formalism of Lim, Prade, and Richard [LPR21], which served as the basis for the approach presented in Chapter 6, with the relational view of APs. In our understanding of the approach, there is indeed an intermediate step between the input and the output of the model: for analogy detection (see Subsections 2.2.1 and 6.2.2), between the quadruple A, B, C, D and the conclusion on whether the AP $A : B :: C : D$ holds or not, the ANNc model considers $R(A, B)$ and $R'(C, D)$, and then checks wether the two relations are conform with each other ($R(A, B) :: R'(C, D)$) which can be seen as checking to which extent $R \approx R'$.

2.4.1 Propositional logic and Boolean Analogical Proportions

Several propositions have been made to express APs using logical formalisms. For instance, some APs have been expressed within the frame of propositional logic [MP09a]. The different propositions result in different models (in the logical meaning) of APs [CL24], some of which are not considered as APs in the axiomatic sense in the categorization by Prade and Richard [PR18]. In Appendix D, we explore the link between the postulates of APs from Table 2.1 and these Boolean models of analogy [BMD07; Cou+17a; Cou+18; CL24; PR18]. Let us consider two examples of APs that are considered differently depending on the model, using 1 for true and 0 for false:

$$0 : 1 :: 1 : 0, \text{ and conversely } 1 : 0 :: 0 : 1.$$

Some of the Boolean models accept the two APs, and consider the negation ($\neg$, defined as $0 \equiv \neg 1,\ 1 \equiv \neg 0$) as an acceptable underlying transformation for APs, while other Boolean models do not.

2.4.2 Kolmogorov complexity approach

The simplicity principle in cognitive sciences [CV03] states that simpler explanations tend to be preferred over more complex ones. Following this principle, the functional reading of AP has been tackled by Cornuéjols [Cor96], Murena, Dessalles, and Cornuéjols [MDC17], Murena, Al-Ghossein, et al. [Mur+20], and Murena [Mur22], using the idea that the simplest transformation f will be considered for the AP $A : f(A) :: C : f(C)$. To compare transformations, the notion of Kolmogorov complexity is used, and the complexity of f is measured as the length of the minimal program that produces $f(A)$ from A. To estimate the Kolmogorov complexity, a set of basic instructions are defined by Murena, Dessalles, and Cornuéjols, and Murena, Al-Ghossein, et al. [MDC17; Mur+20] for APs between character strings, including insertion, deletion, and copy of some characters from A. The shortest sequence of instructions that is able to obtain $f(A)$ from A and $f(C)$ from C is the least complex. In the case of analogy solving (see Subsection 2.2.1), $f(C)$ is unknown so the previous constraint is relaxed: different sequences of instructions are explored to find the shortest program that is able to obtain $f(A)$ from A and is applicable on C, and from the resulting f it is possible to get $f(C)$.

2.4.3 Structure Mapping Theory

SMT [Gen83; GHK01; JSS23; SS22] is an other interesting formalism of AP, in which parallels are made between two conceptual domains. Each domain is described by related concepts, usually in the form of a conceptual graph. For instance, the astronomical domain contains the concepts of the Earth, the Moon, and the gravitational force that makes the Moon orbit around the Earth. The domain of atomic physics may contain the concepts of nucleus, electron, and the electromagnetic force that makes the electron orbit around the nucleus (in Bohr's model). The principle of SMT is to map the two conceptual structures to one another, and doing so creates APs. Our reading of SMT is the relational reading of APs, as a parallel is made between the relation in one domain and in another domain: in the AP "the Earth is to the Moon as a nucleus is to an electron", the orbital relation between the Earth and the Moon, caused by the gravitational force, is compared with the one between a nucleus and an electron, caused by the electromagnetic force.

2.4.4 Case-Based Reasoning

CBR methods are defined in a very general manner in the work of Badra and Lesot [BL23] as methods to "predict the outcome that makes analogical transfer most likely to succeed when the new case is compared to the retrieved cases." Let us dissect this definition. First, a case is a situation (or problem) associated with an outcome (or result or solution). In CBR, a set of cases, called a case base, is used to predict the outcomes for new situations. The case base contains cases for which the situation is associated with the right outcome. From there, the analogical transfer principle in CBR states that similar situations will have similar outcomes. This can be seen as solving a multi-domain AP, as we show in Example 2.4. Alternatively, we can use the analogical inference principle (Equation (2.1)) as was done by Badra and Lesot [BL22; BL23], as we show in Example 2.5. A more detailed introduction to CBR is given in Section 12.1.

Example 2.4: Link between multi-domain APs and CBR

Let us take the setting where situations are animals and outcomes are their classes, knowing that an elephant is a mammal, a whale is a mammal, and a desert turtle is a reptile (the case base). We can find the most similar animal in our case base (the turtle) and, by analogy, consider that the sea snake is a reptile. This corresponds to an AP "reptile is to desert turtle as reptile is to sea snake".

Example 2.5: Link between the analogical inference principle and CBR

Let us take once again the setting where situations are animals and outcomes are their classes, knowing that an elephant is a mammal, a whale is a mammal, and a desert turtle is a reptile (the case base).

One can argue that "an elephant is to a whale as a desert turtle is to a sea snake" holds: elephants and whales are large vertebrates that do not lay eggs, while turtles and snakes are small vertebrates that lay eggs; additionally, elephants and desert turtles live on the ground in arid climates, while whales and sea snakes live in water. By the analogical inference principle, "mammal is to mammal as reptile is to the class of sea snakes", and we can conclude that sea snakes are reptiles.

Chapter 3

Machine learning, deep learning and vectorial representations

Chapter contents

Deep Learning (DL) is a branch of Machine Learning (ML), where the goal is to define and optimize a parametric model for a specific task. These notions are explained in Section 3.1. Usually, the most fitting parameters are the ones resulting in the highest model performance, but other concerns are taken into account, such as the ability to generalize to unseen data, and several notions of fairness and explainability.

In DL, a model follows a structure called the model architecture, which is usually composed of pre-existing building blocks. The building blocks mentioned in this thesis are described in Section 3.2. An important part of tackling tasks using DL is to find the best representation of the manipulated object by the model. In Section 3.3, we describe key methods to obtain this representation, called an embedding.

Once all these fundamentals are explained, we discuss in Section 3.4 some applications of APs (see Chapter 2) in the domain of DL, and in particular APs manipulating embeddings.

For further background, see for instance [24] for ML concepts and [Sar21] for an extensive introduction to DL.

3.1 Terminological distinctions

DL is the field of study of models called artificial Neural Networks (NNs). However, the notion of model is used in many domains with no consensus on the meaning of model and neighboring notions. For instance, as DL is a branch of ML many concepts and methods of the former are the same as the latter. However, a specific terminology and methodology is followed in DL that differs in some points from traditional ML.

To avoid confusion, this section is dedicated to defining the key ML and DL notions that we use in this thesis. This section has no normative value, but reflects the use we make of the terms of the field, hence the lack of specific references. Each key notion will be defined and illustrated with two examples developed throughout this section, which is split in two: Subsection 3.1.1 defines what a parametric model is and Subsection 3.1.2 explains the notions linked to training, *i.e.*, how a model is manipulated in ML and DL.

More technical details are available in Subsection 3.1.2 and Section 3.2.

3.1.1 From the task to the parametric model

Below are defined the notions of *model* (Definition 3.1) and the particular case of *parametric model* (Definition 3.2), that is the core of ML. To the best of our knowledge, the majority of ML approaches consists in choosing or defining a parametric model, then training the model on data to solve a given task. As the majority of our work focusses on ML approaches, when we do not explicitly specify that a model is non-parametric, by "model" we mean "parametric model".

Definition 3.1: Model

A **model** is a mathematical object computing an output from an input. A model can be partially characterized by its input and output.

The output is also called the *prediction* of the model, as a model is typically used to predict. If we simplify, a model is a function that predicts.

Definition 3.2: Parametric model

A **parametric model** is a model where the computation depends on a finite set of values that can be modified, called parameters. In ML, most models are parametric models, and the parameters are *learned* from data during a process called *training*.

When necessary to specify the parameters of a model, we will write them as θ in index of the model.

Definition 3.3: Task

A *task* is a problem to solve. It is usually defined by the input and the expected output of the model.

Defining the task to solve is a very important part of the DL methodology. It is what will determine the input and output of the model, and in turn will define what kind of parametric model can be used to solve the task and what is considered good performance. The definition of the task also impacts what kind of data is used (real numbers, integers, words?). It guides the choice of the model, and tasks are often given names in the literature: to cite only a few, there are classification tasks, regression tasks (Example 3.1), and language modeling tasks like the one in Example 3.2 below. In this thesis we will consider the analogy detection and analogy solving tasks (see Chapter 2).

To be more specific, a possible recipe for tackling a problem with ML is:

1. identify the *problem* to solve;
2. formulate the *problem* as a *task* to tackle;
3. prepare data for the *task*;
4. train and evaluate a parametric model using the data.

Hyperparameters. In DL, model parameters are distinguished from *hyperparameters*, in that hyperparameters are not learned, and they tend to be used to define how the training (see Subsection 3.1.2) will be done or to define the structure and general behavior of the model.

Example 3.1: Meteorologic model

Let us take the task of predicting the average temperature of the next day, given the average temperature of the past 3 days. The input is 3 real-valued numbers (average temperature of the past 3 days) and the output is another real-valued number (temperature of the next day).

Tasks such as this, where a continuous value is predicted, are usually called regression tasks.

A meteorologic model is a mathematical object which, given meteorologic information, *predicts* the weather.

A very simple parametric model we can use to predict temperatures is the weighted sum of three numbers x_1, x_2, x_3: $\text{sum}_\theta(x_1, x_2, x_3) = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$, where x_1, x_2, x_3 are the input of the model and the weights $\theta = \{\theta_1, \theta_2, \theta_3\}$ are its parameters.

For simplicity, let us order x_1, x_2, x_3 such that x_3 corresponds to the day before the day to predict.

To summarize:

- **task:** regression
  - **input:** 3 temperatures
  - **output:** temperature of the next day
- **model:**
  - **input:** three real-valued numbers
  - **output:** a single real-valued number

Example 3.2: Bigram language model

Let us take the task of predicting the next word in a sentence, given the past two words. The input is two words (categorical data) and the output is another word (categorical data).

Tasks such as this, where one or multiple categorical values (labels, classes, elements from a finite set, *etc.*) are predicted, are usually called classification tasks. Such tasks are usually modeled in DL using likelihoods, *i.e.*, probabilities over the possible output values. Then, either the most likely output is chosen, or a sampling process is applied to obtain non-deterministic behavior.

A language model in its simplest form is a mathematical object which, given words previously appearing in a text, predicts the most likely newt words.

A well-known model used for language modelling is the n-gram model. The simplest version of this probabilistic model predicts the conditional probability $p(w|w_1, \ldots, w_n)$ of each $w \in V$ appearing as the next word of the sequence $w_1, \ldots, w_n$, based on the conditional probabilities obtained from the training data. As a parametric model, the conditional probabilities for all the words in V for all the possible n-gram (V^n) are the parameters of the model. The previous words $w_1, \ldots, w_n$ are the input, and the probabilities for each candidate word in V are the output of the model.

To summarize:

- **task:** language modeling
  - **input:** two words
  - **output:** next word
- **model:**
  - **input:** two words

- **output:** probabilities over possible words to be the next word

3.1.2 Training a model

Training, loss, and loss function. To obtain the best possible performance of our model, we need to determine if the model is good or bad at doing the task, with a performance indicator, and change the parameters to obtain better performance. The indicator that we optimize is called the *predictive loss* as, traditionally, it is a measure of the error made by the model between the value it predicts $\widehat{y}$ and the expected value y. The loss function is denoted with $\mathcal{L}_{\text{loss function name}}$, and the loss for a prediction $\widehat{y}$ and an expected value y is written $\mathcal{L}_{\text{loss function name}}(\widehat{y}, y)$. The purpose of training a model f_θ on a dataset D is to solve the optimization problem:

$$\underset{\theta}{\operatorname{argmin}} \sum_{(x,y)\in D} \mathcal{L}_{\text{loss function name}}(f_\theta(x), y), \tag{3.1}$$

in other words, to find the set of parameters that minimizes the loss on the dataset.

Remark 3.1

In DL the loss is by default minimized. To maximize a performance indicator, the inverse of this indicator is used as the loss. As an example, we use this principle in Subsection 12.3.1 to maximize the competence.

In DL, the optimization is usually done in three steps, repeated until some stopping criterion is fulfilled:

1. the loss $\ell = \mathcal{L}_{\text{loss function name}}(f_\theta(x), y)$ is computed for a set of examples $(x, y) \in D$;
2. *backpropagation of the loss* is used to find the gradient (or partial derivative) for each parameter with regards to the loss function, which informs us on which direction to adjust the parameters θ of the model to optimize ℓ;
3. the last step is the update step, where a gradient-based optimization algorithm such as Stochastic Gradient Decent (SGD) or Adam [KB15] is applied to iteratively adjust the parameters based on the value of the gradient.

To compute the gradient of the parameters, the loss function must be differentiable. To the best of our knowledge, DL frameworks (such as PyTorch) use automatic differentiation to perform backpropagation by using the chain rule for derivation. This property of derivatives allows to combine derivatives of elementary operations and functions, which are provided by the DL framework, to obtain the derivative of a bigger function. An overview of modern optimization techniques based on (stochastic) gradient decent is available in [Rud16], and we provide simple examples of the optimization process in Examples 3.3 and 3.4.

Dataset. In ML and DL a task is associated with data, from which we build a set of examples, *i.e.*, a set of inputs associated with the expected output of the model. This data is used to train the model and evaluate its performance.

In DL, datasets are split into three non-overlapping sets of examples:

- the training set contains examples used to train the model;
- the development set contains examples used to make decisions about the model but not directly used to train the model;
- the test set contains examples used to evaluate the performance of the model on unseen data.

We provide example datasets in Examples 3.3 and 3.4.

The non-overlap constraint for the test set allows to observe the performance of the model on data that was not used during training, among other reasons, to confirm that the model will properly generalize to unseen data. For instance, some models become very good at predicting the outcome for the data seen in training, but when confronted with new data they perform perform poorly, like a person learning a lesson by heart without understanding the content.

For the development set, the reasoning is similar. This set can be used, for instance, to decide which variant of the model generalizes best during training, to continue training this variant. Alternatively, by detecting the moment when the performance on the development set starts to drop, it is possible to detect when further training will harm the generalization ability of the model, and therefore prevent overfitting (explained at the end of this subsection). This technique is called early stopping. The development set is also used to compute performance metrics that would be too costly to compute on each example of the training set. As the development set and test set do not overlap, making decisions for the model on the development set does not bias the performance evaluated on the test set.

Example 3.3: Data and training for the meteorologic model

Data. Let us take the following synthetic table of temperatures covering one week.

Day	M.	Tu.	W.	Th.	F.	Sa.	Su.
t	7.5	4.6	4.2	11.9	13.4	10.8	8.8

From this data, we build the following sets of examples, excluding the development set as we will not need it:

- training set: $\{((7.5, 4.6, 4.2), 11.9),\ ((4.6, 4.2, 11.9), 13.4)\}$;
- test set: $\{((4.2, 11.9, 13.4), 10.8),\ (11.9, 13.4, 10.8), 8.8)\}$;

using the $\{(input_i, output_i), (input_j, output_j), \dots\}$ format associating inputs with their expected outputs.

Model training. As the weighted sum (sum_θ) we use for our model is differentiable, we can use gradient-based optimization.

For regression tasks such as our temperature prediction, the Mean Squared Error (MSE) loss between the predicted value $\widehat{y}$ and the expected value y is frequently used:

$$\mathcal{L}_{\text{MSE}}(\widehat{y}, y) = (\widehat{y} - y)^2 \tag{3.2}$$

When considering multiple examples, the MSE is averaged over all the instances.

Let us observe the value of $\mathcal{L}_{\text{MSE}}$ on the test set for two arbitrary parameter configurations of sum_θ:

- using $\theta = \{1/3, 1/3, 1/3\}$ (average of the last 3 temperatures), we have: $\text{sum}_\theta(4.2, 11.9, 13.4) \approx 9.83$ and $\text{sum}_\theta(11.9, 13.4, 10.8) \approx 12.03$, for a total $\mathcal{L}_{\text{MSE}}((9.83, 12.03), (10.8, 8.8)) \approx 5.69$.
- using $\theta' = \{0.2, 0.3, 0.5\}$ (more recent temperature has more weight), we have: $\text{sum}_{\theta'}(4.2, 11.9, 13.4) = 11.11$ and $\text{sum}_{\theta'}(11.9, 13.4, 10.8) = 11.8$, for a total $\mathcal{L}_{\text{MSE}}((11.11, 11.8), (10.8, 8.8)) \approx 4.55$.

Example 3.4: Data and training for the bigram language model

Data. Let us take the following tongue twister as our data: "Can you can a can as a canner can can a can?"

Let us first discard case and punctuation, to obtain the following set of possible words, called the *vocabulary*: $V = \{a, as, can, canner, you\}$.

From this data, we build the following sets of examples, excluding the development set as we will not need it. We use the $\{(input_i, output_i), (input_j, output_j), \dots\}$ format associating inputs with their expected outputs:

- training set: $\{((can, you), can)$, $((you, can), a)$, $((can, a), can)$, $((a, can), as)$, $((can, as), a)$, $((as, a), canner)$, $((a, canner), can)$, $((canner, can), can)\}$;

- test set: $\{((can, can), a),\ ((can, a), can)\}$;

Model training. As we directly use the parameters of the model as output, our 2-gram model is differentiable.

For classification tasks such as our language modelling, the Cross Entropy loss (or *negative log likelihood*) between the predicted probabilities $\widehat{p}$ and the expected probabilities p for each possible class is frequently used:

$$\mathcal{L}_{\mathrm{CE}}(\widehat{p}, p) = -\sum_{w \in V} p(w) \log \widehat{p}(w) \tag{3.3}$$

where $p(w) = 1$ if w is the expected value and $p(w) = 0$ otherwise. In our case for the expected output w, we can simplify $\mathcal{L}_{\mathrm{CE}}(\widehat{p}(w)) = -\log \widehat{p}(w)$. When considering multiple examples, the Cross Entropy is averaged over all the instances.

Let us observe the value of $\mathcal{L}_{\mathrm{CE}}$ on the test set for two parameter configurations of $p_\theta(w_i | w_{i-2}, w_{i-1})$.

- Using the raw probabilities of appearance of a word as the expected output in the training set as the model parameters (0-gram model), we have $\theta = \{\widehat{p}(can) = 4/8,\ \widehat{p}(a) = 2/8,\ \widehat{p}(as) = 1/8,\ \widehat{p}(canner) = 1/8\}$.

  This results in the following loss values: $\mathcal{L}_{\mathrm{CE}}(\widehat{p}(a)) = -\log \widehat{p}(a) = -\log 1/4 \approx 0.602$ and $\mathcal{L}_{\mathrm{CE}}(\widehat{p}(can)) = -\log \widehat{p}(can) = -\log 1/2 \approx 0.301$ for a total of about 0.452.

- Let us now consider the actual 2-gram model. Notice that as (can, can) does not appear in the training set, the conditional probability $\widehat{p}(a|can, can)$ cannot be estimated. Actual n-gram models usually fallback on the $(n-1)$-gram probability, then to the $(n-2)$-gram, *etc.* Therefore, we use the 1-gram to approximate $\widehat{p}(a|can, can) \approx \widehat{p}(a|can) = 1/3$. Considering only values that will appear in the test set, we have $\theta = \{\widehat{p}(can|can, a) = 2/2,\ \widehat{p}(a|can, can) = 1/3, \ldots\}$.

  This results in the following loss values: $\mathcal{L}_{\mathrm{CE}}(\widehat{p}(a|can, can)) = -\log \widehat{p}(a|can, can) = -\log 2/2 \approx 0$ and $\mathcal{L}_{\mathrm{CE}}(\widehat{p}(can|can, a)) = -\log \widehat{p}(can|can, a) = -\log 1/3 \approx 0.477$, for a total of about 0.239.

Batch training. A full iteration of a training algorithm over all the examples in the training set is called an epoch. As mentioned above, in the training algorithms of DL, the loss is computed for a group of examples before updating the parameters of the model. The group of examples is called a mini-batch.

The two extremes of batch training are *(i)* using batches of one example and *(ii)* using a single batch containing all the data of the training set. It has been shown [Ben12; Dek+12; Li+14, among others] that settings closer to *(i)* converge faster but tends to get stuck in local minima, while settings close to *(ii)* guaranty proper convergence in many cases but are slower to converge. Using mini-batches of tens or hundreds of examples results in better convergence speed than accumulating the loss over a full epoch, achieves better generalization, and gets stuck less often in local minima than accumulating the loss for the full training set [WM03, Table 2].

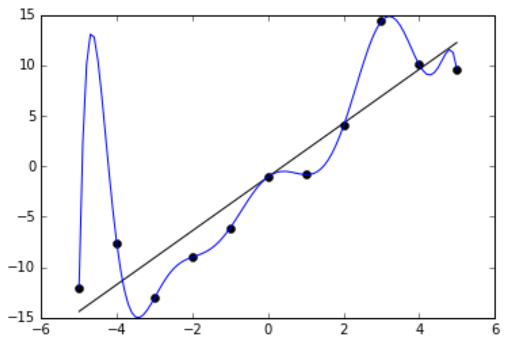
Learning rate. The optimization algorithms used in DL repeatedly update the parameters of the model in the direction opposite to the gradient (as the gradient indicated the direction that increases the parameters). How far the parameters are moved in this direction depends on a training hyperparameter called the *learning rate.*

To explain the impact of learning rate on the optimization process, the landscape defined by the loss function in the parameter space is often compared with an actual landscape [WM03]. In this analogy, the learning rate is the size of the steps a giant would make, always walking in the direction of the downwards slope (the gradient decent). The learning rate impacts the speed of the convergence: smaller learning rates means moving slower towards the optimum, or walking slower in the analogy. It also determines the behavior of the optimization algorithm when there are local minima or narrow valleys. For instance, a small learning rate may result in the parameters getting stuck in a valley of the parameter landscape, where the landscape goes up in all directions despite it not being the lowest point. Conversely, a large learning rate may help the optimization algorithm skip over such valleys, but if the global optimum is located in one such valley it may never be reached.

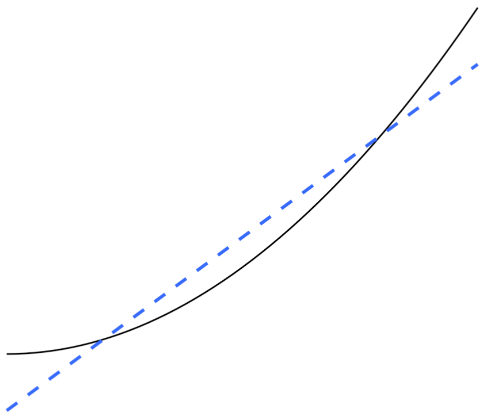
In practice, due to the large number of parameters, there is almost never an actual valley in the parameter landscape and it is rare that the model gets stuck in a local minimum, even with a small learning rate [Ben12]. Additionally, modern gradient decent methods implement a number of optimizations that solve or at least mitigate issues caused by the learning rate [KB15; Rud16].

Overfitting, underfitting, and Vapnik-Chervonenkis dimension (VC dimension). As mentioned before, some models learn the training set by heart without properly generalizing to unseen data. This is called overfitting, and is a common problem in ML, and is related with the ability of a ML to fit complex data. In technical terms, this "learning capacity" can be characterized by the VC dimension [WS22] of the problems it can handle. Overfitting is usually observed when a parametric model has more parameters than necessary to learn to perform the task it is trained for. Taking the example of a classifier, because the model can fit more complex decision boundaries, it will be able to fit even the finer nuances in the training set, in particular the noise, as illustrated in next paragraph. Conversely, underfitting is observed when the model is not complex enough and is unable to learn the decision boundary. Overfitting and underfitting can also be caused by insufficient or poorly distributed data, or by a too long or too short training of the model.

For instance, consider a polynomial that we use to learn a curve in a two-dimension space. If the polynomial as a degree of 1, only lines ($f_\theta(x) = ax + b$ for $\theta = \{a, b\} \in \mathbb{R}^2$) can be learned, and the model will not be able to learn anything of the form $y = x^2$, *i.e.*, we will observe underfitting (see Figure 3.1b). The higher the degree of the polynomial, the more complex a curve can be learned, but if there is even a bit of noise in the data, the model might diverge from the ideal solution by overfitting the data (see Figure 3.1a).



(a) Credits: Ghiles, CC BY-SA 4.0, via Wikimedia Commons



(b) Credits: MohammadMehdiZare, CC BY-SA 4.0, via Wikimedia Commons

Figure 3.1: Examples of overfitting (Figure 3.1a) and underfitting (Figure 3.1b) from Wikimedia.

3.2 Building blocks of Deep Learning models and common models

A NN is a parametric model that is usually composed of multiple interconnected layers. The way these layers are organized and connected is called the architecture of the NN model.

To be more precise, the layers can be seen as the atomic building blocks of complex NN architectures. In this section, we describe several architectures that are used in the approaches described in this thesis. Specific NN architectures can be used as layers for larger NNs, and the name of the architecture is then used as the name of the layer. The architectures we describe in this section are:

- the artificial neuron (or perceptron) and the Multi-Layer Perceptron (MLP) in Subsection 3.2.1;

- the Convolutional Neural Network (CNN) in Subsection 3.2.2;
- the major sequence models, namely: the Recurrent Neural Network (RNN), Long- and Short-Term Memory neural network (LSTM), the mechanism of attention, and the transformer model in Subsection 3.2.3.

3.2.1 The artificial neuron and the Multi-Layer Perceptron

The notions presented in this subsection introduced in by McCulloch and Pitts [MP43]. A more detailed explanation of the principles presented here can be found in Chapter 4 of the open access book written by Nielsen [Nie15].

The basic building block of a NN, and what gave the "neural" in its name, is the artificial neuron. Introduced by McCulloch and Pitts [MP43] as the perceptron, it is a weighted and biased sum of real-valued inputs, passed through an activation function *act*:

$$f_\theta(x_0, \dots, x_n) = act(b + \sum_{i=0}^{n} x_i w_i),$$

for some inputs $x_0, \dots, x_n$, weights $w_0, \dots, w_n$ and a bias b, with the parameters $\theta = \{b, w_0, \dots, w_n\}$. Some frequently used activation functions are the Rectified Linear Unit (ReLU), the sigmoid (σ), and the hyperbolic tangent (tanh) [GBB11]:

$$\begin{aligned} \mathrm{ReLU}(x) &= \max(x, 0), && (3.4)\\ \sigma(x) &= \frac{1}{1+e^{-x}}, && (3.5)\\ \tanh(x) &= \frac{e^x + e^{-x}}{e^x - e^{-x}}. && (3.6)\\ & && (3.7) \end{aligned}$$

If multiple perceptrons are put together, taking the same inputs, they form a layer of neurons called a fully-connected layer (or perceptron layer). This is usually represented using matrices, for n input values and m neurons in the layer:

$$f'_{\theta=\{W,B\}}(X) = act(WX + B)$$

with X the vector of size n, W the weight matrix of size $m \times n$, B the bias vector of size m, and applying the activation function *act* component wise. The output of f'_θ is a vector of size m containing the output of each neuron.

Stacking multiple such layers creates a Multi-Layer Perceptron (MLP), with the outputs of each layer serving as input for the next layer.

The activation function introduces a non-linearity, which is the source of the expressive power of a NN. A specific activation function is usually chosen based on design needs. For instance, applying the sigmoid function on the output is common to perform binary classification as the output is a strictly increasing function of the input, and the image is between 0 and 1 which can serve as the labels of each of the two classes.

The hyperparameters of a fully-connected layer are the number of inputs n, the size of the layer m, and the activation function *act*.

3.2.2 The Convolutional Neural Network layer

The Convolutional Neural Network (CNN) architecture was developed to manipulate images [Den+88; LeC+89], inspired by the neurologic structure of the eye. The workings of the CNN is detailed and illustrated in the work of Dumoulin and Visin [DV16] and animated in the associated repository[1].

As illustrated in Figure 3.2, contrarily to a fully-connected layer where each neuron is connected to all the inputs, in a CNN layer a neuron is only connected to a region of the input, called its perceptive field. The neuron, which is structured as a perceptron with all the values in its perceptive field as input, is called a CNN filter (or kernel), and its size corresponds to the size of the perceptive field.

[1] `https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md`

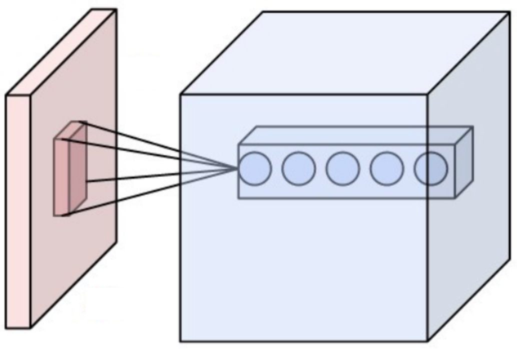


Figure 3.2: Neurons of a CNN layer (blue), connected to their receptive field (red). Credits: Aphex34, CC BY-SA 4.0, via Wikimedia Commons

Among other interesting properties, by applying the same filter (or neurons with the same weights) on different regions of the input, a CNN is able to perform the same processing in all areas of the inputs. It is also able to process inputs of arbitrary size (at least the size of the perceptive field) by applying the filter as many times as necessary, and the size of the output will depend on the size of the input. A pertinent analogy is often made between a magnifier that moves around a picture and CNN filter. How the filter moves across the input is determined by the stride, which is the number of values (or pixels) it moves by in each dimension. For instance, a 2×2 filter with a $(1, 1)$ stride will move by one value each time, and the preceptive field of the filter will overlap with 4 other perceptive fields, one at each corner (except at the border of the input).

A CNN layer may be composed of multiple independent filters, for instance the 5 filters in Figure 3.2. Additionally, using the example of the RGB encoding of the pixels in an image, the input of a filter can be spread across multiple channels. When there are multiple channels in the input, by default each filter uses all the channels as input. For instance, a 2×2 filter on a RGB image wil have a total of 3 (for the channels) $\times 2 \times 2 = 12$ input values. This mechanism is particularly useful to chain CNN layers, as the different filters of one layer wil serve as different channels of the next layer.

When a filter is applied at the border of an input, it is possible to apply the filter only within the boundary of the input. It is also possible to have the filter partially outside the boundaries, by using *padding* which consists in expanding the input beyond its size. Using the analogy of images, this corresponds to adding black pixels around the original input such that we can fit the filter. The value used in the padding area is usually a constant such as 0, but other approaches have been proposed. For example, a 3×3 filter applied without padding on a 5×7 image will result in a 3×5 output (with a $(1, 1)$ stride). As can be seen, the size of the output is smaller than the size of the image, and the values at the very border of the image will not appear in the center of the filter. However, if we pad the input by extending it by one component in all dimensions, with the same filter, stride and input we would obtain a 5×7 output, and all the pixels appear as the center of the perceptive field.

To summarize, the hyperparameters of a CNN layer are, for each dimension (by default 2 dimensions):

- the size of the perceptive field, or filter size along the dimension;
- the stride along the dimension;
- the amount of padding along the dimension.

Other major hyperparameters are the value(s) for padding, the number of input channels and the number of filters, *i.e.*, the number of output channels.

3.2.3 Long- and Short-Term Memory neural network and transformer models

In Natural Language Processing (NLP), the most frequent format of data is the sequence. For instance, a word is a sequence of characters, a sentence or a document are sequences of words,

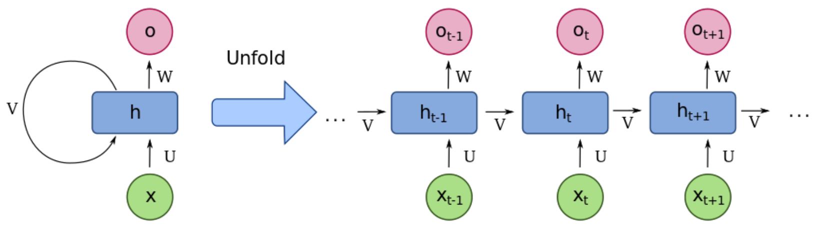


Figure 3.3: Representation of a RNN. Credits: fdeloche, CC BY-SA 4.0, via Wikimedia Commons

speech is a sequence of sound, *etc.*.

In recent years, significant advances have been made in dealing with this kind of data, with the impressive successes of Pretrained Language Models (PLMs) [Ake+19; Dev+19; Wan+20a; Zer+22; ZM20, among many others]. However, the processing of sequences is not a new problem, and a class of models called the Recurrent Neural Networks (RNNs) has been proposed to tackle this kind of input.

The principle of RNN. A sequence of related inputs $x_1, \dots, x_n$ and the corresponding outputs $o_1, \dots, o_n$ can be separated in what are called timesteps (in reference to time series), where all the elements are vectors. The principle behind a RNN is that the output y_t at a timestep t depends not only on input x_t but also on all preceding inputs $x_1, \dots, x_{t-1}$. To model this, the RNN is defined recurrently as follows:

$$h_t = f(x_t, h_{t-1}),$$
$$o_t = g(h_t),$$

where h_t is called the *hidden state*, with a *recurrent unit* f and an *output function* g. For instance, the Elman RNN [Elm90] is defined as follows, for parameters $\theta = \{W_x, W_h, B\}$:

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + B),$$
$$o_t = h_t.$$

This is illustrated in Figure 3.3. To be able to work, a RNN requires an initial hidden state, that can be provided, but is usually initialized to a vector of zeros.

The hidden state h_t contains accumulated information from $x_1, \dots, x_t$, and works as a unit of memory. However, only so much information can be stored in a real-valued vector of finite dimension, and the capacity of the memory is limited. As a consequence, one of the challenges that have been tackled in the evolution of RNNs is how far back does the RNN “remember”, and how to better handle this memory.

The Long- and Short-Term Memory neural network (LSTM). Different functions f, g have been proposed using the principle of RNN. The most frequently used model is the LSTM [HS96], that adresses the limitation of how long information is stored in memory using a long term memory and a short term memory, inspired by human cognition. The short term memory is called the hidden state h_t, and is also used as the model output. The long term memory is called the cell state c_t.

To achieve the long- and short-term memory, three mechanisms are implemented, illustrated in Figure 3.4. They use a combination of sigmoid (to obtain values between 0 and 1) and a component-wise multiplication to create a “gate” that selects if the information will be used (sigmoid closer to 1) or removed from the vector (sigmoid closer to 0), for each component of the multiplied vector. The three mechanisms are:

- a forget gate F_t, to select which part of the previous cell state c_{t-1} is kept in memory;
- an input gate I_t, to select which information from the input and short-term memory will be stored in memory;

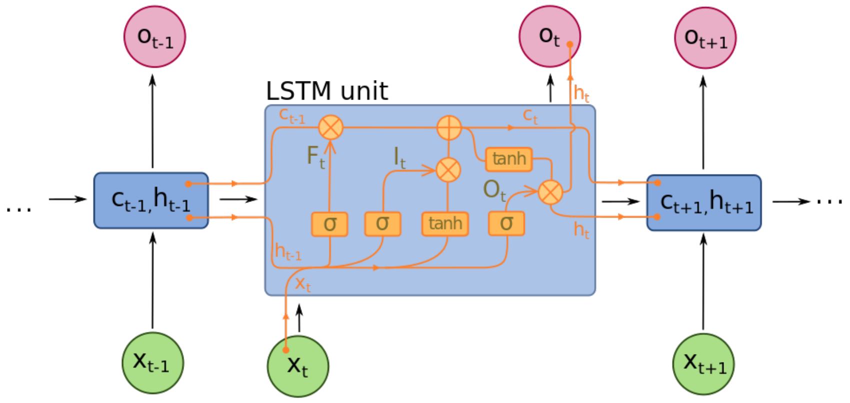


Figure 3.4: Representation of an LSTM. Credits: fdeloche, CC BY-SA 4.0, via Wikimedia Commons

- an output gate O_t, to select which information from the long- and short-term memory will be used for the next hidden state.

The detailed workings of the LSTM are illustrated in Figure 3.4. Formally, using $\times, +$ for the component-wise product and addition, the diagram corresponds to the following formulas:

$$
\begin{aligned}
F_t(x_t, h_{t-1}) &= \sigma(W_{xF}x_t + W_{hF}h_{t-1} + B_F),\\
I_t(x_t, h_{t-1}) &= \sigma(W_{xI}x_t + W_{hI}h_{t-1} + B_I),\\
O_t(x_t, h_{t-1}) &= \sigma(W_{xO}x_t + W_{hO}h_{t-1} + B_O),\\
h'_t &= \tanh(W_{xh'}x_t + W_{hh'}h_{t-1} + B_{h'}),\\
c_t &= (F_t(x_t, h_{t-1}) \times c_{t-1}) + (I_t(x_t, h_{t-1}) \times h'_t),\\
h_t &= O_t(x_t, h_{t-1}) \times \tanh(c_t)
\end{aligned}
$$

with the model parameters $\theta = \{W_{xF}, W_{xI}, W_{xO}, W_{hF}, W_{hI}, W_{hO}, B_F, B_I, B_O, W_{xh'}, W_{hh'}, B_{h'}\}$.

The Bidirectionnal LSTM (BiLSTM). The BiLSTM [GS05] is a variant of the LSTM that reads the elements of the input sequence in the forward and backward order simultaneously. This is achieved by using two independent LSTM layers, one taking the input sequence in the forward direction $(x_1, \ldots, x_n)$ and the other one in the backward direction $(x_n, \ldots, x_1)$.

The Attention mechanism. For an extensive overview of attention, we recommend the blog post from Weng [Wen18].

The mechanism of attention was introduced to address the limitation of memory in automatic translation with RNN [BCB15]. The principle of attention is based on a weight a_t called an attention score, computed for each element x_t in a sequence (or any set of inputs). This weight is high for elements that are useful for the task at hand, and low otherwise. Then, the attention score a_t is applied to select information from the sequence, usually by computing a sum weighted by the attention score, using a similar mechanism as the gates in LSTM:

$$
\begin{aligned}
s &= \sum_{t=1}^{n} \alpha_t x_t\\
\alpha_t = \mathrm{softmax}(a_1, \ldots, a_n)_t &= \frac{e^{a_t}}{\sum_{t'=1}^{n} e^{a_{t'}}}.
\end{aligned}
$$

The softmax applied on a set of input produces for each input an output between 0 and 1, the sum of all outputs being 1.

There exist many different kinds of attention [Wen18], for instance:

- position-based attention [LPM15], where the attention weight depends only on the position t of the element in the sequence;

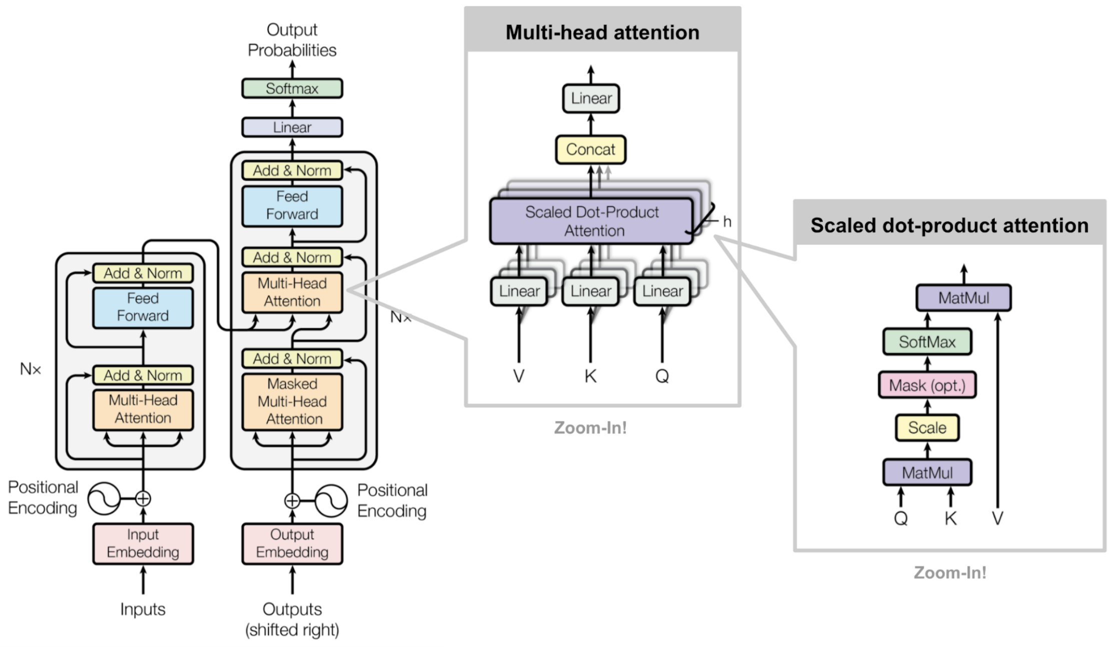


Figure 3.5: Structure of the transformer architecture [Wen18, Figure 17]

- dot product attention [LPM15], that compares x_t with a query q by computing the dot product $x_t \cdot q$;
- scaled dot product attention [Vas+17], which normalizes the dot product attention normalized by $\sqrt{n}$ for a sequence $x_1, \dots, x_n$ to reduce some issues with backpropagation.

When used in RNNs as in [BCB15], attention is used at each timestep to complement information in the output h_t with the summary s of all the inputs, and the query q is the output h_t of the RNNs.

The transformer architecture. Introduced in the work of Vaswani, Shazeer, et al. [Vas+17], the transformer is a sequence model that uses the attention mechanism. It solves the inherent limitations of RNN with regard to the length of the sequence that can be effectively stored in memory, as the model has direct access to all the elements in the input sequence. The structure of the transformer is quite complex, as can be seen in Figure 3.5, so we will not go into the details here and refer to [Vas+17; Wen18] instead. To summarize the architecture, the transformer is composed of two interconnected stacks of attention layers and MLP: an encoder stack, and a decoder stack. The model uses two types of multi-head attention, where the attention is applied multiple times with variants of the input obtained by two layer MLP. The two types of multi-head attention are self-attention, used within each stack between MLPs, and encoder-decoder attention, that connects each layer of the decoder with the corresponding layer in the encoder.

3.3 Embedding models

NNs manipulate real-valued vectors and matrices. A key aspect in DL is to represent the objects manipulated by the NN using such vectors and matrices, called *embeddings*. The performance of DL approaches depends on the quality of the embeddings used, which corresponds to the amount and nature of the information they contain as well as the properties of the vector space in which they are defined (called embedding space). In other words, if the information needed to fulfill a task is easy to access in the embeddings, it is simpler to obtain good performance. This is illustrated in Example 3.5.

To obtain high quality embeddings, it is common to train a NN model that transforms basic representations such as one-hot vectors (see Subsection 3.3.1) into more refined embeddings. This kind of model is called an embedding model. It is possible to train an embedding model together with the model that will use the embeddings on a target task, but it is also possible to pre-train the embedding model on a different task, following a pre-training and fine-tuning workflow (see Subsection 3.3.2). A popular pre-training task is the reconstruction task, that we describe in Subsection 3.3.3, as it does not require labeled data to pre-train the embeddings.

Learning an embedding model of higher quality often requires higher amounts of data (therefore significant training time) and larger VC dimension (therefore larger or more complex embedding models). This can be challenging, as for many applications where the data is very complex but relatively sparse. To circumvent this limitation, large-scale pre-trained embedding models are used. For instance, in NLP, BERT [Dev+19, detained in Subsection 3.3.5] has publicly available model checkpoints[2], that are fine-tuned in a variety of applications. Other popular pre-trained word embedding models are presented in Subsection 3.3.4. The main difference between BERT and the approaches presented in Subsection 3.3.4 is that the latter produces the same embedding for a given word not matter the context the word appears in, while the former produces embeddings that depends on context of use of the word.

Example 3.5: Importance of the information in embeddings

Consider the analogical equation:

$$\text{“}\textit{dog}\text{”} : \text{“}\textit{puppy}\text{”} :: \text{“}\textit{cat}\text{”} : x \text{ that accepts the solution } x = \text{“}\textit{kitten}\text{”}.$$

If we are not aware that a “puppy” is a young “dog” and similarly for “kitten” and “cat”, we will not be able to give the expected solution to the analogical equation.

If we train embeddings e suitable for the task, they should contain information such as being a cat or being a dog, and being an adult or a juvenile. Below are manually crafted embeddings that contain this information and are suitable to solve the analogical equation, with the first component indicating 1 for canine and 0 otherwise, and the second component indicating -1 for young and 1 for adult:

$$\begin{aligned} e(\text{“}\textit{dog}\text{”}) &= [1,1] & e(\text{“}\textit{puppy}\text{”}) &= [1,-1], \\ e(\text{“}\textit{cat}\text{”}) &= [0,1] & e(\text{“}\textit{kitten}\text{”}) &= [0,-1]. \end{aligned}$$

3.3.1 One-hot vectors

A one-hot vector encoding of an element x_i in a set $\{x_1, \ldots, x_n\}$ is a vector of size n containing 0 for all components except component i which is 1. An example is given in Example 3.6.

While one-hot vectors are simple to put in place and unambiguously describe each element in the set, they require as many dimensions as elements in the set which can become an issue when many elements are present (for instance, the many words in a language). One-hot vectors also lose any relatedness between the elements, for instance in Example 3.6, there is no information linking “*dog*” to “*puppy*”, contrary to the handcrafted embeddings in Example 3.5.

Despite these limitations, one-hot vectors are often used as a first representation as input to an embedding model, that will learn to integrate the necessary relatedness information.

Example 3.6: One-hot vector encoding

Consider the analogical equation from Example 3.5:

$$\text{“}\textit{dog}\text{”} : \text{“}\textit{puppy}\text{”} :: \text{“}\textit{cat}\text{”} : x \text{ that accepts the solution } x = \text{“}\textit{kitten}\text{”}.$$

With regard to the set {“*dog*”,“*puppy*”,“*cat*”,“*kitten*”,“*cow*”,“*calf*”,“*bull*”}, the one-hot vec-

[2] A model checkpoint is a configuration of the parameters of a model obtained as the result of training.

tors of the words appearing in the analogical equation are:

$$\begin{aligned} e(\text{``\textit{dog}''}) &= [1,0,0,0,0,0,0], \\ e(\text{``\textit{puppy}''}) &= [0,1,0,0,0,0,0], \\ e(\text{``\textit{cat}''}) &= [0,0,1,0,0,0,0], \\ e(\text{``\textit{kitten}''}) &= [0,0,0,1,0,0,0]. \end{aligned}$$

3.3.2 The pre-training and fine-tuning workflow

Compared to training an embedding model from scratch on a target task, the pre-training and fine-tuning workflow consists in:

1. **pre-training:** train a model on one or more tasks that cover the information required to solve in the target task;
2. **fine-tuning:** use the pre-trained model as part of another model for the target task, and use task-specific data to train the task-specific model together with the pre-trained model.

The pre-training is often done on tasks where data is more easily available than for the target task. For instance, BERT is trained on a variety of general domain NLP tasks, and doing so, the model learns to encode general purpose information in the embeddings. There exist variants of this workflow, for instance few-shot learning [Wan+20b] where the fine-tuning step is reduced to a minimum or even removed in zero-shot applications [Yas+24].

The pre-training and fine-tuning workflow has many advantages. First and foremost, the pre-training process brings the embeddings closer to an optimal configuration as they already contains some, if not all, the information necessary to solve the target task. As the embeddings is closer to an optimal state, less training data (and training time) is necessary compared to training from scratch, to obtain equivalent performance on the target task. Second, for large and popular models such as BERT, the result of pre-training is usually made available publicly. As many people reuse the pre-trained embedding models and perform only the fine-tuning, there is a factorisation of the pre-training costs. From there, it becomes reasonable to have larger scale embedding models and pre-training, resulting in higher quality embeddings and more performant downstream system.

3.3.3 The Auto-Encoder

The Auto-Encoder (AE) architecture was introduced by Kramer [Kra91] and can be seen as a lossy compression algorithm, composed of an *encoder* and a *decoder*. The encoder compresses the objet to represent o into an embedding $e(o) = z$, usually with a small dimension compared to the objet encoding. Then, the decoder decompresses the embedding back into an object $d(z) = \widehat{o}$. If we were to train an AE to represent the words in Example 3.6, the encoder compresses the information of a word into the embedding and the decoder decompresses the embedding back into a word.

In the reconstruction task, the AE encodes and decodes the objects to represent and is trained to minimize the difference between the original object o and the reconstructed object $d(e(o))$. In our compression algorithm analogy, this corresponds to minimizing the compression loss. Because the decoder is able to recreate the original object from the embedding, the latter contains the key information to represent the former. In more details, the decoder will learn to reproduce systematic or redundant parts of the training data without relying on the embedding, while key information to differentiate the training examples are encoded in the embedding by the encoder.

Note that the transformer architecture introduced in Subsection 3.2.3 is an AE.

One major advantage of the reconstruction task is that is does not require any labeled data, as the input is also the target output. A variant of the reconstruction task is the denoising task, where a noisy version of the object is used as input and the AE must reconstruct the original version of the data. Similarly, the masking task hides parts of the object instead of adding noise [Bae+22; Dev+19]. This helps the AE learn to rely on the non-noisy (or non-hidden) parts of the input to predict the noisy (or hidden) parts.

3.3.4 Distributional semantics and word embedding models

Representing word semantics is of key interest to perform many NLP tasks, from understanding to generating text. Many approaches have been developed to produce embedding spaces to represent word semantics. In this subsection, we give a brief overview of four major approaches to represent individual words, or as we will explain in next paragraph, individual tokens. In Subsection 3.3.5, we present another embedding approach called BERT that accounts for surrounding tokens when computing the embedding. These approaches encode what is called the distributional semantics of the word, as they used based on the assumption that words with similar meanings will appear in similar contexts. This is called the distributional hypothesis.

Tokenization. The first thing when processing a sentence or a document, is that from the perspective of an algorithm, textual data is represented as a string of characters. The question of how to cut the string into meaningful units, such as words or punctuation, is answered by the tokenization process, that cuts the text into tokens. Similarly to a morpheme that is a minimal subword with an associated meaning, a token is a string of characters that is considered meaningful.

Tokenization can go from a simple split around punctuation and spaces to more refined processes that cut larger words, such as "tokenization", into the more frequently seen substrings "token" and "-ization".

Latent Semantic Analysis (LSA). LSA [Dum+88] is an early word embedding technique, based on the term-document occurrence matrix, that counts the number of occurrences of each words from a vocabulary in a corpus of documents. Words with similar meanings should appear in similar documents, following the distributional hypothesis, and have similar rows in the matrix.

By applying a matrix factorization method, the term-document matrix is compressed along the dimension corresponding to the documents, by collapsing similar documents into more general concepts. The resulting vectors represent the distributional semantics of each word, and similar vectors correspond to words with similar distributions.

Global Vectors (GloVe). GloVe [PSM14] is a method that uses a word-word co-occurrence matrix instead of the term-document matrix of LSA. To obtain the matrix, a context window is used to identify the words that appear in the direct vicinity of other words. The GloVe embeddings are trained using a regression approach to obtain a factorisation of the word-word co-occurrence matrix.

Word2Vec. The Word2Vec model was proposed by Mikolov, Kai, et al. [Mik+13] as two variants of the same intuition: the Skip Gram and Continuous Bag of Words (cbow). Let us consider a word w_i in a context window $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$ of size 5. The cbow method trains an embedding model e to estimate the conditional probability $p(w_i|w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2})$ with a function f defined in [Mik+13] taking as input the embeddings of $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$. The name cbow comes from the fact that the bag containing the words occurring in the vicinity of w_i is used to predict the embedding of w_i. The Skip Gram method takes the opposite approach, and learns to predict the conditional probabilities $p(w_{i-2}|w_i), p(w_{i-1}|w_i), p(w_{i+1}|w_i), p(w_{i+2}|w_i)$ from the embedding of w_i and the embedding of the word to predict.

FastText. The FastText model proposed in [Boj+17] uses the same Skip Gram and cbow to train embeddings. The main difference is that instead of taking whole words as tokens as done in Word2Vec, FastText uses the embeddings for the n-grams composing a word to obtain the embedding of the word. Taking an example from [Boj+17], for 3-grams, "where" is decomposed in "<wh", "whe", "her", "ere", "re>", and the full word "<where>", with $<, >$ special characters marking the start and end of the word respectively. These special characters allow to identify n-grams which appear at the boundaries of word.

3.3.5 Bidirectional Encoder Representations from Transformer

Bidirectional Encoder Representations from Transformer (BERT), introduced by Devlin, Chang, et al. [Dev+19], is a transformer model where only the encoder stack is used, therefore only self-attention is applied. As BERT accounts for the surrounding context when embedding each token, it is called a contextualized embedding model.

Model input. The input of BERT is text, typically a pair of sentences (or spans of text) or a single sentence (or span of text). For exemple, such pairs of sentences can be a question and a paragraph in which to find the answer (Extractive Question Answering (Ex-QA) task), or two sentences that we want to know if the former implies the latter on a semantic level (textual entailment task).

This text is split into tokens represented using the WordPiece technique [Wu+16], that consists in identifying frequent sub-words in a manner similar to identifying morphemes. The sub-words are obtained by iteratively finding sub-words that frequently appear together (above a certain frequency threshold), merging them, and adding them to the vocabulary of known sub-words. In the case of BERT, a vocabulary of 30 000 WordPiece tokens are used. Among these tokens, ponctuation is also included.

Additionally, the input of BERT contains special tokens:

- `[CLS]`: a special token at the beginning of any input sequence, called the classification token, the embedding of which is used as input for classification tasks in the pre-training of the model;

- `[SEP]`: it defines the boundary of a particular segment of the model, for instance, the context and question in Ex-QA;

- `[MASK]`: used in the masked language modelling task described further below.

To handle textual information, it is very important to be aware of the order of the words as can be seen in Example 3.7. The transformer layers are based on the attention mechanism, that is applied identically on all tokens in the sequence, therefore no positional information is modeled by the attention itself. To tackle this issue, BERT (and other transformer models) uses a mechanism called positional encoding, which encodes both the relative and absolute position of each input token. BERT also uses segment encoding, which is a special embedding added to every token embedding when two sentences are input, with a different segment embedding for the first and second sequence.

Example 3.7: Importance of order information in textual data

The sentences “the cat hunts the mouse” and “the mouse hunts the cat” describe opposite situations. If we discard word order, they are both equivalent to “cat hunts mouse the the” and the differences in meaning between the two sentences disappear.

Pre-training tasks. Two pre-training tasks are used for BERT, that can be seen as reconstruction tasks.

One of them is a masked token prediction task called *masked language modelling*, where some tokens of the input are hidden or replaced by random tokens, and the embeddings of the masked tokens are used to train a classifier to predict the original tokens. The classifier is a single fully-connected layer with as many outputs as tokens in the vocabulary, combined with a softmax to obtain a probability distribution over all tokens.

The other task is *next sentence prediction*, where two sentences from the dataset are used as input, and a binary classifier predicts whether the sentences are consecutive or not. The classifier is a single perceptron with sigmoid activation, that predicts 0 for not-consecutive and 1 for consecutive.

The first task allows to train the embeddings to contain distributional information, as the model learns which tokens are likely to appear in place of the masked token in the context of the input text. The second task guides BERT to encode relational information between the two input sentences, useful as a starting point for the Ex-QA or textual entailment tasks, among other. Both pre-training tasks are applied simultaneously, meaning that a pair of sentences with masked/replaced tokens is used as input, and the token prediction and next sentence prediction model are used respectively on the embeddings of the masked words and of `[CLS]`.

3.4 Approaches to APs in DL

3.4.1 Methods to manipulate APs in embedding spaces

In Chapter 2, we introduced the notion of AP, where the features of objects are compared to check if we have a relation of the form "A is to B as C id to D". As mentioned in Subsection 3.3.4, embeddings can be seen as vectors where each component represents a feature, such as a semantic feature in the case of semantic embeddings. As such, embeddings have been widely used in analogy detection and analogy solving.

Analogy preservation and embedding models. Manipulating APs in embedding spaces is based on a rarely expressed postulate, mentioned for instance by Hwang, Grauman, and Sha [HGS13]: that the APs on the object manipulated are reflected in the embedding space. This corresponds to the analogical inference principle introduced in Subsection 2.2.1, and can be rewritten as inference rules:

$$\frac{A : B :: C : D \quad \text{holds}}{e(A) : e(B) :: e(C) : e(D) \quad \text{holds}} \quad \text{and} \quad \frac{e(A) : e(B) :: e(C) : e(D) \quad \text{holds}}{A : B :: C : D \quad \text{holds}}, \tag{3.8}$$

for $e(A), e(B), e(C), e(D)$ the embeddings of A, B, C, D.

In other words, we assume the implication:

$$A : B :: C : D \quad \text{holds} \quad \Longleftrightarrow \quad e(A) : e(B) :: e(C) : e(D) \quad \text{holds}. \tag{3.9}$$

For analogy solving in embedding spaces, the process can be split into three steps: project the analogical equation in the embedding space (3.10), solve the analogical equation in the embedding space (3.11), and project the solution back in the object space (3.12). In Equations (3.10) to (3.12), we represent the three steps by putting on left the object space and on the right the embedding space:

$$\begin{aligned} A : B :: C : x \quad \text{holds} \quad &\Longrightarrow \quad & e(A) : e(B) :: e(C) : y \quad \text{holds} \qquad &(3.10)\\ & & y = e(D) \qquad &(3.11)\\ A : B :: C : D \quad \text{holds} \quad &\Longleftarrow \quad & e(A) : e(B) :: e(C) : e(D) \quad \text{holds}. \qquad &(3.12) \end{aligned}$$

An important challenge for analogy solving is therefore to find an analogy preserving embedding model, ideally for which we know the inverse to compute $D = e^{-1}(y)$.

The parallelogram rule. In Subsection 2.3.1 Equation (2.5), we mentioned the parallelogram rule, which is a generalization of the arithmetic proportion (Equation (2.4)) to multi-dimensional data. It can be expressed with different formulas, for instance:

$$\begin{aligned} \vec{A} - \vec{B} &= \vec{C} - \vec{D} \\ \vec{A} + \vec{D} &= \vec{B} + \vec{C} \\ \vec{D} &= \vec{C} - (\vec{A} - \vec{B}) \\ \vec{D} &= \vec{C} + \vec{B} - \vec{A}. \end{aligned}$$

The parallelogram rule can be applied on embeddings, as can be seen in Example 3.8.

Example 3.8: Example of the parallelogram rule

Taking the manually crafted embeddings from Example 3.5, with the first component indicating 1 for canine and 0 otherwise, and the second component indicating −1 for young and 1 for adult:

$$\begin{aligned} e(\text{“}\textit{dog}\text{”}) &= [1, 1] \quad & e(\text{“}\textit{puppy}\text{”}) &= [1, -1] \\ e(\text{“}\textit{cat}\text{”}) &= [0, 1] \quad & e(\text{“}\textit{kitten}\text{”}) &= [0, -1]. \end{aligned}$$

We are able to check that the following AP holds:

$$\text{“}\textit{dog}\text{”} : \text{“}\textit{puppy}\text{”} :: \text{“}\textit{cat}\text{”} : x, \quad x = \text{“}\textit{kitten}\text{”}$$
$$e(\text{“}\textit{dog}\text{”}) - e(\text{“}\textit{puppy}\text{”}) = [0, 2] = e(\text{“}\textit{cat}\text{”}) - e(\text{“}\textit{kitten}\text{”}).$$

We are also able to solve the analogical equation:

$$\text{“}dog\text{”} : \text{“}puppy\text{”} :: \text{“}cat\text{”} : x$$

$$\begin{aligned} e(x) &= e(\text{“}cat\text{”}) - (e(\text{“}dog\text{”}) - e(\text{“}puppy\text{”})) \\ &= [0, 1] - ([1, 1] - [1, -1]) \\ &= [0, -1] = e(\text{“}kitten\text{”}). \end{aligned}$$

This parallelogram rule has been used since the first works on AP [RA73], and it is a key element in the methodology employed by semantic embedding approaches [DGM16; Dum+88; Mik+13]. Indeed, being able to use a simple formula like the parallelogram rule to compare and transfer semantic relations in the embedding space, is an indication that the semantic relations in question are encoded in the embedding space.

3CosAdd and 3CosMul. To solve analogical equations in modern embedding models such as Word2Vec, GloVe, or FastText (see Subsection 3.3.4), two of the most used methods are 3CosAdd [Mik+13] and 3CosMul [LG14]:

$$\text{3CosAdd}D = \underset{\widehat{D}}{\operatorname{argmax}} \cos(e(\widehat{D}), e(B) - e(A) + e(C)) \tag{3.13}$$

$$\text{3CosMul}D = \underset{\widehat{D}}{\operatorname{argmax}} \frac{\cos(e(\widehat{D}), e(B))\ \cos(e(\widehat{D}), e(C))}{\cos(e(\widehat{D}), e(A)) + \varepsilon}. \tag{3.14}$$

These two approaches implicitly generate a solution y and approximate $e^{-1}(y)$ by retrieving the closest candidate $\widehat{D}$ from the vocabulary, based on the value of its embedding $e(\widehat{D})$. 3CosAdd can be seen as the parallelogram rule to generate y followed by cosine similarity to recover the closest existing solution $e(\widehat{D})$. On an intuitive level, 3CosMul is similar to 3CosAdd, except using a geometric proportion ($\frac{A}{B} = \frac{C}{D}$) on the angles of A, B, C with regards to $\widehat{D}$. To avoid a 0 denominator if $\widehat{D}, A$ are aligned with regards to the origin, an ε (small) is added in (3.14). As 3CosMul is harder than 3CosAdd to fully grasp intuitively, we refer the reader to [LG14, page 175] for a detailed description.

Limitations of fixed formulas. Chen, Peterson, and Griffiths [CPG17] and Rogers, Drozd, and Li [RDL17] argue that 3CosAdd and 3CosMul significantly differ from human performance, and even more so for the parallelogram rule. To solve this limitation, several approaches have proposed to leverage examples of APs to train models or embeddings. For instance, Lim, Prade, and Richard [LPR19] proposed to learn two layer NNs for analogy detection and analogy solving on pre-trained embeddings, detailed in Section 6.2. Other approaches propose to learn embeddings that perform well in terms of AP [DZF19; GDM16; Kar+18], or learn linear transformations of the embedding space while relaxing the formulation of the parallelogram rule [BJS18].

3.4.2 Applications of APs in embedding spaces

The above-mentioned methods to manipulate APs have been applied in different domains.

Applications to word semantics. Word semantics is one of the best known application of APs on embedding spaces. To the best of our knowledge, first applications date from the LSA [Dum+88] and vector decomposition methods, the precursors of modern word embeddings. For instance, APs have been used by Mikolov, Kai, et al. [Mik+13] to show the interest of Word2Vec, one of the earliest *trained* word embedding model, and revealed that a model trained for distributional semantics encoded semantic regularities that can be found by the parallelogram rule [MYZ13]. Word2Vec, GloVe, and vector decomposition methods were compared in the work of Drozd, Gladkova, and Matsuoka [DGM16], using several variants of the retrieval method 3CosAdd.

Several authors, including Chen, Peterson, and Griffiths [CPG17] and Rogers, Drozd, and Li [RDL17], have identified limitations in the performance of traditional distributional semantic word embedding models, but also of the datasets used to evaluate the analogy solving performance of the models. The latter limitations led to the production of several datasets, for instance the Bigger Analogy Test Set (BATS) [GDM16] and the Japanese Bigger Analogy Test Set (JBATS) [Kar+18].

As mentioned at the end of previous subsection, Lim, Prade, and Richard [LPR19; LPR21] proposed an approach to perform analogy detection and analogy solving on word embeddings obtained with GloVe. One particularly interesting aspect of this work is that a data augmentation process based on the postulates of APs was used to make the model conform with the postulates, and to generate non-APs for analogy detection. We describe this data augmentation process in Section 6.3.

Applications to images. While the idea of learning models on pre-trained word embedding for APs was implemented by Lim, Prade, and Richard [LPR19], similar ideas have been implemented for instance for image retrieval [SZF15] and image generation [Ree+15]. In that line of work, a dataset [Bit+23] was recently proposed for analogy solving by retrieval, offering analogical equations between images including carefully selected distractors (images in the set of retrieval candidates but different from the expected candidate). In the same article, Bitton, Yosef, et al. propose several baselines, including the parallelogram rule applied on the embedding space, and the parallelogram rule applied on situation prediction features extracted with a pre-trained model.

Applications to sentences. In the work of Taillandier, Wang, and Lepage [TWL20], APs are used to express semantic relations between sentences. The authors use the analogical inference principle (see Subsection 2.2.1) to translate the fourth element D of an AP $A : B :: C : D$ in another language, leveraging the analogical equation $f(A) : f(B) :: f(C) : x$ formed by the translations of the other three elements A, B, C. The model they train performs alignments between sentences, in a process reminiscent of attention mechanisms, and the alignment is used to generate the translation $f(D)$. Other approaches have been developed to solve analogical equations directly between sentences using sequence models (see Subsection 3.2.3) and the principle illustrated in Equations (3.10) to (3.12) [ML23; WL20].

In the work of Yasunaga, Chen, et al. [Yas+24], analogical prompting was used to guide PLMs in answering questions properly, by asking the PLM to provide examples of the question. This kind of process is particularly interesting, as it operationalizes analogy solving. First, the model is asked to find a source ratio $Q_s : A_s$ suitable for the task at hand, and then it performs analogy solving to produce the answer A_t to a target question Q_t:

$$Q_s : A_s :: Q_t : x \text{ solved by } x = A_t.$$

Other approaches have been developed to explore the analogy detection and analogy solving capacity of PLMs [SS22; Ush+21].

Wang and Lepage [WL20] proposed a generation framework to solve the semantic and structural APs between on phrases proposed in [Lep19]. They use an AE sequence model based on RNN (see Subsection 3.2.3) trained to reconstruct sentences, and perform simple arithmetic operations on the embedding space to solve analogical equations, including the parallelogram rule. Once the analogy between embeddings is solved, the decoder is used to generate the solution from the predicted embedding, and fulfill the purpose of the inverse embedding e^{-1} mentioned in the beginning of Subsection 3.4.1. The use of a generative model achieves significantly better results than using a basic retrieval model, namely a k-Nearest Neighbors (k-NN) model applied on the same embedding space.

Applications to word morphology. In the work of Mikolov, Yih, and Zweig [MYZ13], the authors identified that some grammatical regularities are captured in the representations obtained with Word2Vec, and can be identified using the parallelogram rule. These grammatical regularities imply morphological regularities. In particular, Cotterell, Schütze, and Eisner [CSE16] were able to use a Gaussian graphical model to model morpho-grammatical changes in the embedding space of Word2Vec. To the best of our knowledge, beyond the work of Mikolov, Yih, and Zweig, no DL approach has been proposed to tackle APs specifically in word morphology. To fill this gap, we propose the ANN framework in Part II using a similar approaches to the ones in the works of Lim, Prade, and Richard and Wang and Lepage [LPR19; LPR21; WL20] for word and sentences semantics. More details on non-DL approches to APs morphology are presented in Subsection 4.3.2, and DL approches to morphology outside of analogical considerations in Subsection 4.3.1.

Applications to structured and semi-structured data. Structured and semi-structured data, such as knowledge graphs (structured) or a combination of text and tabular data (semi-structured), have also seen applications of AP through DL.

For instance, Jarnac, Couceiro, and Monnin [JCM23] proposed an analogy detection approach in knowledge graphs, using the same ANNc model used in our work [Als+21a, see also Subsection 6.2.2] and in the work of Lim, Prade, and Richard [LPR19; LPR21]. The approach was applied for knowledge graph pruning, a subtask of automatic knowledge graph construction where the entities (nodes of the graph) in the neighborhood of a seed entity $\mathbf{e}_s$ are filtered based on how relevant they are for a target application. The approach of Jarnac, Couceiro, and Monnin predicts if a node $\mathbf{e}_r$ of the knowledge graph should be pruned with regards to $\mathbf{e}_s$ using an AP $\mathbf{e}_s : \mathbf{e}_r :: \mathbf{e}'_s : \mathbf{e}'_r$ where the decision to prune $\mathbf{e}'_r$ when $\mathbf{e}'_s$ is the seed is known from manual pruning performed by experts. If the decision represented by the ratios $\mathbf{e}_s : \mathbf{e}_r$ and $e'_s : e'_r$ are the same, then the AP is classified as valid.

The works of Alsaidi, Couceiro, et al. [Als+22a; Als+22b] use the ANNc model and analogical data augmentation of the ANN framework (see Section 6.3) to check if two patient stay records p_1, p_2 belong to the same patient, and infer which of the two corresponds to an earlier stay than the other. To do so, the analogy detection model was trained using APs involving the pair $p_1 : p_2$ and a pair where the relation is known. Zervakis, Vincent, et al. [Zer+22, see also Chapter 10] also use ANNc and analogical data augmentation (see Section 6.3) to preform TSV (see Section 10.1), a task involving several types of textual data.

APs and Siamese architectures in NNs. In DL, a Siamese architecture is an architecture that contains a model f' that is reproduced twice (or more) in a larger architecture f, and the output of f' is joined to compute the final output of f. For instance, if it is possible to represent the architecture of a model with two inputs, written $f_\theta(x, y)$, can be formulated as $f_\theta(x, y) = g_{\theta^3}(f'_{\theta^1}(x), f'_{\theta^2}(y))$, then f may be called a Siamese architecture.

This kind of architecture is particularly appropriate to manipulate APs, as it aligns well with the relational reading $R(A, B) :: R(C, D)$. This idea has been implemented by sharing the parameters θ^1, θ^2 of the Siamese part [SZF15] or not [LPR19; LPR21].

Chapter 4

Word morphology

Chapter contents

In this chapter, we give a brief introduction on morphology and morphological transformations in Section 4.1. We then give some first examples of APs in morphology, and highlight two important links between analogy and morphology in Section 4.2. Finally, in Section 4.3, we present a variety of computational approaches to morphology and APs on morphological relations.

4.1 Brief introduction to word morphology

Different levels of study of words in linguistics. In the field of linguistics, language is studied at different interdependent levels. If we focus on words, one can study for instance: the meaning of words (semantics), the organization of words in sentences (syntax), the pronunciation of words (phonology and phonetics), or the structure of words (morphology), on which we focus in Part II and in this chapter.

Morpheme, root, and lemma. The basic unit of morphology is called a morpheme. It is usually defined as the smallest meaningful unit of meaning a word can be cut into. For instance, “unlearned” can be cut into “un-”, “learn”, and “-ed”. The respective meanings are “not”, “learn”, and “past participle”. Morphemes are categorized depending on if they can be used alone (free morphemes) or if can only be used as a part of a word (bound morphemes). In the previous example, “learn” is free while “un-” and “-ed” are bound.

Morphemes that carry the core of the meaning of a word (usually free morphemes) are called the root or stem of the word: “linguist” can be split into the root “lingu-” and the suffix “-ist”. Notice that a root might correspond to an existing word (“learn”) or not (“lingu-”).

A similar notion is that of lemma, which is the canonical form of a word. In some cases, the lemma and root are identical: “linguists” can be split into the root “linguist” and the suffix “-s”, with the lemma “linguist”. In other cases, they may differ: “linguist” can be split into the root “lingu-” the suffixes “-ist”, while the lemma is “linguist”.

Inflectional morphology and derivational morphology. Morphology is usually separated into inflectional morphology and derivational morphology. On the one hand, derivational morphology refers to morphological transformations that allow to create new words in a systematic manner, for instance in English the prefix "un-" allows to create "unaware" from "aware" in the same manner as "untold" from "told". On the other hand, inflectional morphology describes morphological transformations that express a change in the grammatical nature of a word without altering the core meaning of the word. For example, "looked" is the result of a change of tense in the English verb "(to) look" by adding the suffix "-ed". Generally, derivational morphology produces distinct lemmas, while inflectional morphology does not.

Some morphological transformations. There exist a wide variety of morphological transformations, among others:

- the most well known for English speakers are *prefixation* and *affixation* in which a morpheme is attached at the beginning or the end of the word, respectively; for instance, "un-" is a prefix while "-ed" is a suffix;
- with *infixation*, a morpheme is added inside a word, for instance in French, "boiter" (meaning "to limp") supports the infix "⟨-ill-⟩", to become "boitiller" (meaning "to limp a little");
- the principle of *reduplication* is to repeat a word or part of it, potentially with a small change, for instance in Japanese, "hito" (meaning "a person") becomes "hitohito" (meaning "people");
- in *simulfixation*, one or more parts of a word is replaced, for instance, "mouse" becomes "mice".

Some morphological transformations are dependent on the rest of the word, such as the suffix "-s" for plural in English that behaves differently in "cat" ("cat*s*") and in "bus" ("bus*es*"). Some morphological transformations such as *reduplication* are typically context dependant, while some other become context dependent due to language-specific phenomena. For instance, in Finnish, there is a mechanism of vowel harmony [RH99], where the vowels are divided into neutral (i and e), and two group of harmonic vowels: front (y, $ö$, $ä$) and back vowels (u, o, a). Front and back vowels do not appear together in native Finnish non-compound words, therefore a morpheme might have to be adapted to fit this vowel harmony. Taking an example from [RH99], the essive case suffix "-na/-nä" agrees with the vowels in the root: "pouta" ("dry weather") becomes "pouta-na" while "pöytä" ("table") becomes "pöytänä".

In addition to these, some words evolve with usage, giving birth to exceptional morphological transformations. For example, euphony is, in very simple terms, a change of sound to make a word easier to pronounce, such as "*far*" that becomes "*further*" when adding the suffix "*-ther*".

4.2 Morphological analogy, paradigm tables and morphological innovation

In this section, we first give practical examples of what morphological APs are, in Subsection 4.2.1. Then, we explain the notions of analogical grids and analogical innovation in word formation, respectively in Subsections 4.2.2 and 4.2.3. The link between analogical grids, analogical innovation, and the formalism of APs is discussed in Subsection 4.2.4.

4.2.1 Morphological APs

A morphological AP, is, quite straightforwardly, an AP where the underlying relationships are of morphological nature.

While they are not strictly identical, morphological APs correspond in a lot of cases to grammatical APs or to semantic APs, as can be seen in Example 4.1. This can be the case for inflectional morphology and derivational morphology alike, even if inflectional morphology will be more varied in terms of grammar, and derivational morphology in terms of semantics.

Example 4.1: Morphological, grammatical, and semantic AP

If we consider the AP:

$$\text{“}\textit{undercooked}\text{”} : \text{“}\textit{cooked}\text{”} :: \text{“}\textit{undertrained}\text{”} : \text{“}\textit{trained}\text{”}$$

under the lens of morphology, we can get:

$$\text{“}\textit{under-cook-ed}\text{”} : \text{“}\textit{cook-ed}\text{”} :: \text{“}\textit{under-train-ed}\text{”} : \text{“}\textit{train-ed}\text{”}$$

with the morphological transformation modeled by “*under-cook-ed*” : “*cook-ed*” being: removing the prefix “under-”. Now, through the lens of semantics, we get:

“X has been cooked, but not enough” : “X has been cooked”
:: “X has been trained, but not enough” : “X has been trained”

with the transformation being: going from “X has been Y, but not enough” to “X has been Y”.

Let us consider another AP:

$$\text{“}\textit{cook}\text{”} : \text{“}\textit{cooked}\text{”} :: \text{“}\textit{train}\text{”} : \text{“}\textit{trained}\text{”}$$

under the lens of morphology, we can get:

$$\text{“}\textit{cook}\text{”} : \text{“}\textit{cook-ed}\text{”} :: \text{“}\textit{train}\text{”} : \text{“}\textit{train-ed}\text{”}$$

with the morphological transformation modeled by “*cook*” : “*cook-ed*” being: adding the suffix “-ed”. Now, through the lens of conjugation, we get:

lemma of “(to) cook” : preterit of “(to) cook” :: lemma of “(to) train” : preterit of “(to) train”

with the transformation being: going from the lemma to the preterit.

To manipulate morphological APs, it is only necessary to know of the morphology of the language, even if the semantics of specific words are unknown, as can be seen in Example 4.2. As such, morphological APs are fundamentally close to APs between strings of symbols, as detailed in Subsection 4.3.2.

Example 4.2: Morphological AP without knowledge of semantics

Take the noun “Balrog”, that is exclusive to the universe of *The Lord of the Rings* (to the best of our knowledge). Anyone who does not have knowledge of the universe of *The Lord of the Rings* will not know what “Balrog” means, but will still be able to solve the following analogical equation at the morphological level:

$$\text{“}\textit{cat}\text{”} : \text{“}\textit{cats}\text{”} :: \text{“}\textit{Balrog}\text{”} : x, \quad x = \text{“}\textit{Balrogs}\text{”}.$$

4.2.2 Paradigm tables and analogical grids

A paradigm table (or inflectional paradigm) is a table that lists all inflected forms for some lemma, as described in Sig16[1]. An example taken from [FL18] is given in Table 4.1.

Paradigm tables are strongly linked with morphological APs. In [FL16; FL18], they are put in parallel with analogical grids, the latter being “a [(not necessarily dense)] matrix of words, where four words from two rows and two columns form an AP” [FL18]. An example is given in Table 4.2. More formally, if we write G_r^c the component at row r and column c of the grid, then we have $G_r^c : G_r^{c'} :: G_{r'}^c : G_{r'}^{c'}$ for any two rows r, r' and any two columns c, c' in the grid. According to the authors, the main difference between the two is that paradigm tables are the result of linguistic study, while analogical grids are produced from data.

Remark 4.1

The notion of analogical grid is dependent on the idea that Transitivity holds. Otherwise, $G_r^c : G_r^{c'} :: G_{r'}^c : G_{r'}^{c'}$ and $G_r^{c'} : G_r^{c''} :: G_{r'}^{c'} : G_{r'}^{c''}$ could hold without $G_r^c : G_r^{c''} :: G_{r'}^c : G_{r'}^{c''}$ holding in the same grid, which goes against the very definition of an analogical grid.

[1]https://sigmorphon.github.io/sharedtasks/2016/

	Infinitive	Preterit	Past participle	Present participle
Regular verb	walk smoke	walked smoked	walked smoked	walking smoking
Irregular verb	write think	wrote thought	written thought	writing thinking

Table 4.1: Paradigm table example for english verbs. [FL18, Figure 2]

walk	:	walk*s*	:	walk*ing*	:	walk*ed*
show	:	show*s*	:	show*ing*	:	show*ed*
open	:	open*s*	:	open*ing*	:	
study	:		:	study*ing*	:	
play	:		:	play*ing*	:	play*ed*

Table 4.2: Analogical grid example for english verbs. Missing values correspond to words not present in the dataset. [FL18, Figure 2]

4.2.3 Analogical innovation

As illustrated in Example 4.3, analogy is a powerful tool to generate new but understandable words, using known words as a reference.

Example 4.3: Analogical innovation

Let us consider the following analogical equation:

$$\text{“}small\text{”} : \text{“}smallest\text{”} :: \text{“}best\text{”} : x, \quad x = \text{“}{*}bestest\text{”}.$$

The solution to this analogical equation is agrammatical (by convention in linguistics, marked with an asterisk), but is easy to understand from a morphological point of view. The semantic associated to the suffix “-est” in “small-est” can then be transferred to “best-est”.

The book by Mattiello [Mat17] gives an extensive overview of the mechanisms of analogy in word formation, and gives a more precise description of analogical word creation in their Section 1.3:

> “a new word is coined that is either based on a precise actual model word, or obtained after a set of concrete prototype words which share the same formation (*i.e.* series) or some of their [roots] (*i.e.* word family).” [Mat17]

4.2.4 Analogical innovation and analogical grids

The two notions of analogical innovation and analogical grids are tightly linked. Indeed, analogical grids will typically group together words produced through analogical innovation from the same series or word family. Conversely, let us assume we have a non-dense analogical grid, as in Table 4.2, and that the missing parts are caused by words that do not exist in the language (contrary to Table 4.2). In that case, it is possible to fill the gaps in the grid by analogical innovation as we did for “Balrog” in Example 4.2.

These two notions are strongly related with the AP formalism, as analogy detection is necessary to build analogical grids, and word creation through analogical innovation is a particular application of analogy solving, with the solution not yet existing in the lexicon.

4.3 Approaches to automatic morphology and morphological analogy

The following two subsections describe approaches to automatically perform morphological transformations (Subsection 4.3.1) and to automatically detect APs or solve analogical equations (Subsection 4.3.2).

We distinguish the approaches based on how they were initially formulated. Nevertheless, under the right circumstances, approaches to automatic morphological analogy can be used for inflectional morphology, and conversely. For instance, an analogy solving approach can be used to perform inflectional morphology if given examples of the desired morphological transformation. Once the underlying morphological features in the first ratio of a morphological AP have been identified, an automatic inflectional morphology approach can be used to solve an analogical equation.

4.3.1 Approaches to automatic morphology

Automatic inflectional morphology. Significant efforts have been made in recent years to perform inflectional morphology and derivational morphology automatically. A major player in this field has been the ACL Special Interest Group on Computational Morphology and Phonology (SIGMORPHON) community, which has hosted a number of workshops[2] since 2008 and shared tasks[3] since 2016.

In the most recent shared task in inflectional morphology at the time of writing, namely Sigmorphon 2023 Task 0 (Sig23) [Gol+23], approaches are split into DL approaches and non-DL approaches. DL approaches [CH23; Gir23] use sequence models such as LSTM and transformer to learn how transform a given lemma into a specified inflected form, by taking the lemma and a set of morphological feature as input and generating the inflected form corresponding to the features. Non-DL approaches [Cot+17; KHW23] include methods that estimate prefixation and suffixation rules from candidate prefixes and suffixes obtained by aligning words at the character level. The rules can be obtained with simple comparisons of the differences in the beginning and end of words [Cot+17] or using non-DL ML techniques such as finite state transducers [KHW23] to model more complex transformations.

Similar approaches can be found in previous inflectional morphology SIGMORPHON shared tasks, such as the ones of 2016 [Cot+16] and 2019 [McC+19] among others, but also for other morphology tasks such as morpheme segmentation [Bat+22].

Representing morphology in deep learning To complement research on semantic word embedding techniques, morphological information has been widely considered. First, accounting for characters is useful to be able to handle unseen words, that may come from not being represented in the data, or from being neologisms. Second, morphemes carry meaning (it is even how they are defined, see Section 4.1), and being aware of the morphological structure of a word can help efficiently representing its meaning.

The Ph.D. thesis of Vania [Van20] is a recent and detailed overview on the topic. Among well known embedding approaches, FastText [Boj+17] uses all sequences of adjacent characters of length n, *i.e.*, character n-grams. The more recent BERT [Dev+19] considers WordPiece tokenization, which focusses on frequently observed n-grams, with lengths that might vary. Other models use more elaborate processes to better incorporate morphological processes [AAB20; CR16; LSM13; NR22], sometimes to study morphology itself [Chu+19].

4.3.2 Approaches to automatic morphological analogy

Approaches to algorithmically process morphological analogies have been focused on the formal characterization of APs (see Chapter 2), and we present such approaches in this subsection.

Hofstadter's micro-world. To study the properties of APs on character strings, Hofstadter and Mitchell introduced in [HM95] an experimental setting that is usually called *Hofstadter's micro-world*. This setting contains analogical equations where the elements are strings of letters, and the transformations use relations between the position of letters in the character string (first, last, *etc.*), and relations based on alphabetic ordering (alphabetic successor, alphabetic predecessor). Other relations include the copy of groups of characters, as described in [MDC17], and closely relate to *reduplication* (see Section 4.1).

[2]https://sigmorphon.github.io/workshops/
[3]https://sigmorphon.github.io/sharedtasks/

Example 4.4: Simple example of Hofstadter's micro-world

A well known example of Hofstadter's micro-world, taken directly from [HM95], is the following:

$$\text{“}abc\text{”} : \text{“}abd\text{”} :: \text{“}ijk\text{”} : x.$$

Possible solutions include:

- $x =$ “*ijl*”, by considering that “the rightmost letter was replaced by its alphabetic successor” [HM95];
- $x =$ “*ijk*”, by considering that “*c*” is replaced by “*d*” (and we have no “*c*” in “*ijk*”);
- $x =$ “*ijd*”, by considering that the last letter is replaced by “*d*”;
- $x =$ “*abd*”, by considering that the whole sequence is replaced by “*abd*”.

Symbolic and data-driven learning approaches. Hofstadter's micro-world has been extensively studied to develop approches to APs on strings of symbols in general, as the key requirements are informations on the position, and several relations between symbols defined by experts.

In fact, to the best of our knowledge, all approaches to morphological APs, excluding the one developed in Part II, follow a similar setting, with a set of expert designed operations [LYZ09; Mur+20], relations between characters [HM95], or morphological features [FL18].

In contrast, many of the recent approaches in Subsection 4.3.1 are data-driven models, that rely on data and not experts to learn the operations, relations between (groups of) characters, and features necessary to apply morphological transformations. Our approach, the ANN framework detailed in Part II, is a data-driven approach to morphological APs.

Symbolic approaches to APs. Fam and Lepage [FL18] gather algorithms from their previous work [FL16; Lep98; Lep14], based on postulates proposed by Lepage and Ando [LA96]. Their approach detects and solves morphological APs based on distances between words, using manually designed features such as their length or the occurrence of letters and of specific patterns. In the work of Fam and Lepage [FL16; FL18], the approach was used to generate analogical grids (see Subsection 4.2.2). This approach, that we call Lepage's Nlg toolkit (Nlg) in reference to the name of the Python library made available by Fam and Lepage [FL18], is further detailed in Subsection 7.2.1.

The Alea approach by Langlais, Yvon, and Zweigenbaum [LYZ09] is based on a reformulation from Stroppa and Yvon [SY04; Yvo03] of the edit distance algorithm introduced by Lepage [Lep98]. In practice, this reformulation uses random slicing and merging of the character strings A, B and C to obtain potential solutions to $A : B :: C : x$, and ranks the solutions based on their likelihood of appearance by repeatedly applying the random generation process in a Monte Carlo setting. Additional details on the method are given in Subsection 7.2.2. Good results can be obtained with 1000 repetitions [LYZ09].

A more empirical approach, that we coin Kolmo, was proposed by Murena, Al-Ghossein, et al. [Mur+20]. Following preliminary evidences that humans may follow a simplicity principle when solving analogies [CV03; CA98; MDC17], the authors propose to solve analogical equations $A : B :: C : x$ by finding the x that minimizes the total description length (or Kolmogorov complexity) of $A : B :: C : x$. The total description length is evaluated using a simple description language for character strings and an associated binary code. More recently, an extension of the Kolmo approach was proposed by Murena [Mur22] to measure how transferable the ratio $A : B$ is to a new element C, tackling the problem of the existence of a solution for an analogical equation $A : B :: C : x$. The description language and the optimization problem are detailed in Subsection 7.2.3.

Both Alea and Nlg were shown in the experiments of Murena, Al-Ghossein, et al. [Mur+20] to be outperformed by Kolmo on Kakenhi 15K00317 word analogies from Sigmorphon 2016 Task 1 (Kakenhi-Sig16) [Lep17].

Other approaches have been proposed in the literature, for instance the ones of Neuvel and Fulop [NF02] and Hathout [Hat08] that aligns the occurrences of characters in pairs or quadruples of words, with some similarities with the method of [FL18; Lep98]. Neuvel and Fulop and Hathout have used their respective approaches to extract morphemes and other morphological information from data.

Data-driven learning approach. As mentioned before, to the best of our knowledge and at the time of writing, the approach we propose and develop in our work [Als+21a; Als+21b; Als+21c; Cha+22; Mar+22a; MC24; Mar+22b] that we present in Part II is the only DL approach designed to tackle morphological APs. Note however that this is not the only data-driven learning approach to APs in general, with for instance the approaches we describe in Section 3.4.

Part II
Morphological analogical proportions

This part describes the main work done on the Analogy Neural Network framework (ANN framework). To do so, Chapter 5 introduces the Siganalogies dataset used throughout this part, as well as some running examples taken from Siganalogies. The ANN framework itself is described in Chapter 6, and extensive experimental results are presented in Chapters 7 and 8. Finally, in Chapter 9, we summarize the contributions presented in this part and explain how the ANN framework and Siganalogies are made available following the principles of open science, including interactive demos available to the general public. This part covers contributions published in [Als+21a; Als+21b; Als+21c; Cha+22; Mar+22a; MC24; Mar+22b].

Chapter 5

The Siganalogies dataset of morphological analogies

Chapter contents

To develop and evaluate the performance of the ANN framework on morphological APs, we designed the Siganalogies dataset [Mar+22b]. This dataset contains morphological APs in more than 80 distinct languages and is built upon three datasets: Sigmorphon 2016 Task 1 (Sig16) [Cot+16], Sigmorphon 2019 Task 1 (Sig19) [McC+19], and the Japanese Bigger Analogy Test Set (JBATS) [Kar+18]. The dataset is called Siganalogies: *analogies* is self explanatory, and *Sig* comes from the SIGMORPHON shared tasks Sig16 and Sig19 from which all but one language (Japanese) were extracted.

Each dataset contains words linked by morphological transformations, that we use to create APs as explained in Section 5.1. We survey Sig16, Sig19, and JBATS in Section 5.2, and provide detailed statistics on Siganalogies in Section 5.3. We summarize the tools and features provided with Siganalogies in Section 5.4. A description of the dataset with extensive statistics is also available on the GitHub page of the dataset[1]. We conclude this chapter with a discussion on the limitation of the Siganalogies dataset in Subsection 5.5.3, followed by some perspectives for improvements of the dataset.

5.1 Building APs from morphological transformations

Sig16, Sig19, and JBATS contain triplets $\langle A, B, f\rangle$ with A, B a pair of words related by a morphological transformation f. We write $f = \{\text{feature}_1 = \text{value}_1, \text{feature}_2 = \text{value}_2, \dots\}$ the transformation f described by a set of grammatical features feature$_1$, feature$_2$, that respectively become value$_1$, value$_2$. In the triplet $\langle A, B, f\rangle$, B is the word obtained after applying the morphological transformation f on A. Examples 5.1 and 5.2 are two examples of triplets:

[1] `https://github.com/EMarquer/siganalogies/blob/main/siganalogies_description.pdf`

Example 5.1: Triplet in English

Taking the example from the authors of the Sig16 shared task, going from "*run*" to "*running*" corresponds to a transformation from the lemma to the present participle of "*(to) run*". The corresponding triplet would be:

$$\langle A = \text{“}\textit{ran}\text{”}, B = \text{“}\textit{running}\text{”}, f = \{\text{tense} = \text{present participle}\}\rangle.$$

Example 5.2: Actual triplet from the Finnish data from Sig16

In the Finnish data from Sig16, we have the triple:

$$\langle A = \text{“}\textit{lenkkitossut}\text{”}, B = \text{“}\textit{lenkkitossuilla}\text{”}, f = \{\text{pos} = \text{N}, \text{case} = \text{ON+ESS}, \text{num} = \text{PL}\}\rangle.$$

The morphological transformation here is quite specific: the transformation corresponds to the nominative to essive cases (case = ON+ESS) of a noun (pos = N) for the plural (num = PL). The encoding of morphological features used here is UniMorph, which is used in Sig16 (see Section 5.2) and uses grammatical information to encode the features.

A morphological AP is a quadruple $A : B :: C : D$ where the morphological transformation from A to B is the same as the one from C to D (see Section 4.2): "A is to B as C is to D in terms of morphological transformation". As the morphological transformation is made explicit in the data, we can use two triplets $\langle A, B, f\rangle, \langle C, D, f\rangle$ sharing the same morphological transformation to create the morphological AP $A : B :: C : D$, or $A : f(A) :: C : f(C)$ in terms of the functional view of analogy. As f is not explicit in $A : B :: C : D$, we say it is the underlying transformation of the AP $A : B :: C : D$. The main difference between Sig16, Sig19, and JBATS is the way the triplets $\langle A, B, f\rangle$ are stored (see Section 5.2), which means we can use the above-mentioned process for all datasets. For instance, we can take Example 5.2 and another triplet from the data to create the AP in Example 5.3.

Example 5.3: AP from the Finnish data from Sig16

In the Finnish data from Sig16, we have another triple:

$$\langle C = \text{“}\textit{alko}\text{”}, D = \text{“}\textit{alkoilla}\text{”}, f = \{\text{pos} = \text{N}, \text{case} = \text{ON+ESS}, \text{num} = \text{PL}\}\rangle.$$

From this triplet and the one of Example 5.2, we create the AP:

$$\text{“}\textit{lenkkitossut}\text{”} : \text{“}\textit{lenkkitossuilla}\text{”} :: \text{“}\textit{alko}\text{”} : \text{“}\textit{alkoilla}\text{”}.$$

Remark 5.1

Siganalogies is meant to be used with the data augmentation process described in Section 6.3. For each two pairs, we only generate one AP, *i.e.*, if we generate $A : B :: A' : B'$ we do not generate $A' : B' :: A : B$ as it will be generated by the data augmentation process.

Remark 5.2

APs of the form $A : B :: A : B$ are not automatically produced by the data augmentation process (see Section 6.3), but are generated by our AP building process, as the set of features is the same ($f = f$). We find important to integrate such examples as they are related to the Reflexivity of Conformity postulate, and to Identity, by Exchange of the Means.

5.2 Source datasets

As mentioned at the beginning of this chapter, Siganalogies is built upon Sig16, Sig19, and JBATS, each containing triplets $\langle A, B, f\rangle$ that we use to create our APs.

The Sigmorphon shared tasks[2] are a series of shared tasks focusing on tackling morphology-

[2] https://sigmorphon.github.io/sharedtasks/

related tasks on multiple languages, organized by the ACL Special Interest Group on Computational Morphology and Phonology (SIGMORPHON). For Siganalogies, we consider only the first edition of the shared task (Sig16) and the forth edition (Sig19).

Our main reason for using Sig16 is comparability: it is the dataset used to extract the APs in [Lep17; Mur+20]. To this dataset we first added JBATS to explore Japanese and its very different take on morphology compared to European languages, then we added Sig19 for the large amount of multilingual data it offers.

5.2.1 Sigmorphon 2016

The Sigmorphon 2016 Task 1 (Sig16) [Cot+16] shared task, subtitled "Morphological Reinflection", was organized from 1st December, 2015 to 28th April, 2016. As the name indicates, the focus is on morphological reinflection, which consists in applying an inflectional morphological transformation on a word that may already be the result of such a transformation (hence *re* inflection). Taking Example 5.1 from the authors of Sig16, going from "*ran*" to "*running*" corresponds to a transformation to the present participle applied on "*ran*", which is already an inflected form of "*(to) run*" to some person of the past.

Sig16 proposes the 3 tasks below, in which a word and a set of target features are provided, and the inflected form of the word corresponding to the features must be found.

1. Task 1 focuses on inflection: a lemma is provided, and the task is to transform this lemma following morphological features. In terms of triplets from Section 5.1, we have:

$$\langle A = \text{``}\textit{run}\text{''}, B = \text{``}\textit{running}\text{''}, f = \{\text{tense} = \text{present participle}\}\rangle.$$

2. In task 2, the first version of the reinflection task, two words are provided, both inflected forms of the same lemma. For both words, the corresponding morphological features are provided: instead of triplets, we have quadruplets:

$$\langle A = \text{``}\textit{ran}\text{''}, f_A = \{\text{tense} = \text{past}\}, B = \text{``}\textit{running}\text{''}, f_B = \{\text{tense} = \text{present participle}\}\rangle.$$

3. Task 3 is the "unlabled" version of the reinflection task, which is the same as task 2 but without information about A, *i.e.*, without f_A. In other words, we obtain triplets like

$$\langle A = \text{``}\textit{ran}\text{''}, B = \text{``}\textit{running}\text{''}, f = \{\text{tense} = \text{present participle}\}\rangle.$$

   The main difference between tasks 1 and 3 is that A is a lemma in task 1 and an inflected form in task 3.

For the Siganalogies dataset, we consider only data from Task 1, as was done in [Lep17] and later in [Mur+20].

Remark 5.3

It is possible to apply the same process we apply on task 1 to tasks 2 and 3 to obtain more varied morphological APs.

Most of the data of Sig16 is extracted from the English edition of Wikitionary[3], with the exception of Maltese data which came from the Ġabra open lexicon [Cam13]. The procedure described in [Kir+16] was used for the extraction and the verification of the data, and the morphological features are encoded using the UniMorph Schema [Syl+15]. As mentioned in [Cot+16], the words are written using the corresponding native script, except for Arabic for which the romanized[4] forms available in Wikitionary are used. Wikitionary is crowd-sourced and may contain errors (many of which have probably been corrected since the creation of Sig16), and these errors may exist in the data as no manual checking was done.

[3]`https://en.wiktionary.org`

[4]Romanization is the process of writing using the roman alphabet words of a language that does not use the roman alphabet.

5.2.2 Kakenhi 15K00317 word analogies from Sigmorphon 2016 Task 1

Sig16 is used by Lepage [Lep17] and later by Murena, Al-Ghossein, et al. [Mur+20] as a source for morphological APs, using the same processed we use for Siganalogies but with a slightly different data augmentation process: Lepage only considers APs obtained from Symmetry of Conformity and Inversion of Ratio. The data extracted by Lepage is available online, as the Kakenhi-Sig16 dataset[5]. When comparing the APs from Kakenhi-Sig16 with the ones we extracted directly from Sig16, we noticed some differences reported in [Als+21a]. In particular, accounting for the different data augmentation processes, neither ours nor Lepage's set of APs completely contains the other. The full extent of the difference is reported in Table 5.1, by specifying the coverage rate of one set by the other: for instance, an AP $A : B :: C : D$ from [Lep17] is covered by our version of Sig16 if $A : B :: C : D$ or any of its 8 permutations (see Section 2.3) is present in our dataset.

Language	**Coverage by Lepage's version of ours**	**Coverage by our version of Lepage's**
Arabic	26.73	60.26
Finnish	21.51	92.30
Georgian	68.22	78.51
German	68.07	92.38
Hungarian	33.14	37.39
Maltese	16.70	61.82
Navajo	7.09	10.18
Russian	25.81	91.81
Spanish	54.23	91.44
Turkish	29.12	71.80

Table 5.1: Table 3 from [Als+21a]. Original caption: "Coverage (in %) between Lepage's version of Sigmorphon2016 and the training set of our version. The coverage of the test set is 0% in both directions for all languages and was not included."

5.2.3 Sigmorphon 2019

The Sigmorphon 2019 Task 1 (Sig19) [McC+19] shared task, subtitled "Crosslinguality and Context in Morphology", was organized from 21st December, 2018 to 30th April, 2019. It focusses on the idea of universal morphological inflection, and features "nearly 100 distinct languages"[6]. This dataset is of particular interest for us due to it richness in terms of represented languages: the more languages we can experiment on, the more general our analysis of the ANN framework can be.

Similarly to Sig16, Sig19 proposes 3 tasks: a cross-lingual transfer task (task 1), a task to leverage context in the inflexion process (task 2), and an open challenge for submissions using the data of the shared task (task 3).

From Sig19, we used the data from task 1 to build Siganalogies. This task consists in transferring the inflection mechanisms from a high-resource language (*i.e.*, a language for which data is readily available, such as French, English, *etc.*) to a low-resource language (*i.e.*, a language for which data is hard to come by, for instance, languages with few speakers, local dialects, or non-written languages). 100 language pairs covering 79 unique languages are proposed for this task, with pairs of very related languages (*e.g.*, German and older dialects of German) and distantly related or unrelated pairs (*e.g.*, Greek and Bengali). The distinction between high- and low-resource languages comes only from the amount of data proposed by the dataset, with languages usually considered as high-ressource appearing as low resource in Sig19, for instance, Greek, Russian and Portuguese. For each language, Sig19 contains pairs of words and the corresponding morphological features. In practice, a lemma from the low-resource language and a set of target features are provided, and the corresponding inflected form must be found. To tackle the task, systems are expected to leverage the larger amount of data from the high-resource language.

As in Sig16, the data is from the English Wikitionary, using the updated procedure associated with UniMorph 2.0 [Kir+18], with the exception of four langages:

[5] `http://lepage-lab.ips.waseda.ac.jp/en/projects/kakenhi-15k00317/`, "Experimental data" tab, section "Words: SIGMORPHON data set".

[6] `https://sigmorphon.github.io/sharedtasks/2019/`

> " The Basque language data was extracted from a manually designed finite-state morphological analyzer [Ale+09]. Murrinhpatha data was donated by John Mansfield; it is discussed in [Man19]. Data for Kurmanji Kurdish and Sorani Kurdish were created as part of the Alexina project [WS10; WSF10]. " [McC+19]

Unlike in Sig16, the Arabic data in Sig19 always uses the original writing system. Additionally, some minor fixes on the extracted data are made by the organizers of the Sig19 shared task.

In addition to the large amount of data and vast language coverage, we chose the task 1 from Sig19 to explore if the ANN framework could tackle this cross-lingual transfer task. Indeed, we could use APs $A : B :: C : D$ where A, B come from the high-resource language and C, D from the low-resource language, with A, B selected to match the features specified in the task. Then, the value of D could be found by analogy solving. While this task is not tackled in this thesis, the results described in Chapter 8 are encouraging for further work in the direction of cross-lingual APs.

5.2.4 Japanese Bigger Analogy Test Set

The Japanese Bigger Analogy Test Set (JBATS) [Kar+18] contains analogies in Japanese and was designed based on the Bigger Analogy Test Set (BATS), a dataset of analogies in English designed to be more extensive and balanced than its predecessors. The two datasets contain 4 categories of linguistic relations: derivational morphology, inflectional morphology, lexicographic semantics, and encyclopedic semantics.

Japanese Bigger Analogy Test Set (JBATS) was initially designed to evaluate the performance of the sub-character and character level embedding models in Japanese, and allows for multiple possible writings when necessary. For instance, for the second word (B) of each triplet $\langle A, B, f\rangle$ in the morphological data, a kanji (ideogram-based writing) form and an hiragana/katakana form (syllable-based alphabet writing) are provided to cover possible alternate writings.

As we work on morphological APs, we consider only the derivational and inflectional morphology data, with the subcategories detailed in Table 5.5 and Section 5.3.

5.3 Quantitative information and practical details

This section details the languages available in Siganalogies. We also provide some distributional statistics on morphological features and on the amount of APs that can be extracted for each language.

In total, 82 distinct languages are covered by Siganalogies, counting the 10 languages from Sig16, the 44 high-resource and 44 low-resource languages from Sig19 and the Japanese from JBATS. For the full list of languages covered by Sig16 see Tables 5.2 to 5.4, for high-resource languages from Sig19 see Table 5.6. From Sig19 task 1 we use only the high-resource languages in our experiments, so only the corresponding statistics are provided here. For low-resource languages, refer to the description available on the GitHub page of the dataset[7].

Among the languages of Sig16, 7 are available as high-resource languages of Sig19 (Arabic, Finnish, German, Hungarian, Russian, Spanish, and Turkish) and 2 as low-resource languages of Sig19 (Maltese and Russian). Note that Russian appears as both a high- and a low-resource language in Sig19. In fact, the following languages are available as both high- and low-resource languages in Sig19: Bengali, Czech, Greek, Irish, Latin, Portuguese, Russian, Sorani, and Swahili.

All low-resource languages of Sig19 have less than 25000 APs in each set, with less than 900 APs in the training set. Except Basque and Uzbek, which have 43754 and 7312 APs respectively, all high-resource languages have at least 133000 APs. The set 42 languages (high resource except Basque and Uzbek) will be the one considered for our experiments.

In Tables 5.2 to 5.4 and 5.6, we report three statistics:

- *# APs* is the number of distinct APs we obtain from the procedure described in Section 5.1, which is then multiplied during data augmentation (see Section 6.3).
- *# Features with APs* is the number of distinct features in the dataset that are involved in at least one AP. This value is an indicator of how rich the APs are in terms of morphological

[7]https://github.com/EMarquer/siganalogies/blob/main/siganalogies_description.pdf

Language	# Analogies	# Features with analogies (% of all features)	# Words with analogies (% of vocabulary)
Arabic	373240	220 (98.65%)	13773 (99.97%)
Finnish	1342639	94 (98.95%)	22057 (99.99%)
Georgian	3553763	90 (100.00%)	14587 (100.00%)
German	994740	97 (98.98%)	17307 (99.99%)
Hungarian	3280891	85 (98.84%)	17279 (99.99%)
Maltese	104883	2419 (75.97%)	19338 (95.38%)
Navajo	502637	42 (77.78%)	4502 (99.80%)
Russian	1965533	80 (96.39%)	18793 (99.97%)
Spanish	1425838	83 (98.81%)	17145 (99.99%)
Turkish	606873	179 (95.72%)	14223 (99.94%)
Japanese	26410	20 (100.00%)	1573 (100.00%)

Table 5.2: Statistics of languages from Sig16, for the training data. Languages in bold are also present in Sig19.

features. For this measure, we count each distinct value of each feature separately, as for num and its values PL and SG in Example 5.4 below.

Example 5.4

Let us consider the morphological transformations:

$$\begin{aligned} f_1 &= \{\text{pos} = \text{N}, \text{case} = \text{ON+ESS}, \text{num} = \text{PL}\} \\ f_2 &= \{\text{pos} = \text{N}, \text{num} = \text{SG}\} \\ f_3 &= \{\text{pos} = \text{V}\} \end{aligned}$$

If f_1, f_2 correspond to triplets used to create APs (*i.e.*, f_1, f_2 appear in at least 2 distinct triplets) while f_3 does not, we have 4 features with APs ($|\{\text{pos} = \text{N}, \text{case} = \text{ON+ESS}, \text{num} = \text{PL}, \text{num} = \text{SG}\}|$) which corresponds to 80% of all features.

This value is an indicator of how rich the APs are, in terms of morphological features.

For each language, we also specify the coverage by APs of the features appearing for the language (*% of all features*).

- *# Words with APs* is the number of distinct words in the dataset that are involved in at least one AP.

In Table 5.6, some languages have under 100% *# Features with APs*. This is due to some features (*e.g.*, 25.67% of features for Asturian) that appear in only one pair of words, and therefore cannot appear in an AP as we need at least two pairs of words with the same features to create APs. This effect could be mitigated by using a more relaxed expression of morphological transformations, as discussed in Subsection 5.5.3.

Language	# Analogies	# Features with analogies (% of all features)	# Words with analogies (% of vocabulary)
Arabic	7671	218 (99.09%)	2638 (99.89%)
Finnish	22837	73 (84.88%)	3070 (99.16%)
Georgian	67457	48 (70.59%)	2716 (99.27%)
German	17222	97 (98.98%)	2888 (99.93%)
Hungarian	70565	76 (92.68%)	3517 (99.80%)
Maltese	3775	585 (39.24%)	2288 (67.20%)
Navajo	33976	42 (91.30%)	1578 (99.81%)
Russian	32214	77 (100.00%)	2898 (100.00%)
Spanish	25590	83 (100.00%)	2836 (100.00%)
Turkish	11518	160 (95.81%)	2691 (99.56%)

Table 5.3: Statistics of languages from Sig16, for the development data. Languages in bold are also present in Sig19.

Language	# Analogies	# Features with analogies (% of all features)	# Words with analogies (% of vocabulary)
Arabic	555312	220 (97.35%)	15996 (99.96%)
Finnish	4691453	95 (100.00%)	37857 (100.00%)
Georgian	8368323	90 (100.00%)	19722 (100.00%)
German	1480256	98 (98.99%)	19954 (99.99%)
Hungarian	66195	78 (95.12%)	3448 (99.88%)
Maltese	3707	597 (39.62%)	2315 (67.53%)
Navajo	4843	35 (83.33%)	618 (99.36%)
Russian	6421514	80 (96.39%)	29868 (99.99%)
Spanish	4794504	83 (100.00%)	28230 (100.00%)
Turkish	11360	161 (96.41%)	2675 (99.59%)

Table 5.4: Statistics of languages from Sig16, for the test data. Languages in bold are also present in Sig19.

	Morphological transformation	**Example**	**Pairs**
Inflectional morphology	verb_dict - mizenkei01	会う → 会わ/あわ	50
	verb_dict - mizenkei02	出る → 出よ/でよ	51
	verb_dict - kateikei	会う → 会え/あえ	57
	verb_dict - teta	会う → 会っ/あっ	50
	verb_mizenkei01 - mizenkei02	会わ → 会お/あお	50
	verb_mizenkei02 - kateikei	会お → 会え/あえ	57
	verb_kateikei - teta	会え → 会っ/あっ	50
	adj_dict - renyokei	良い → 良く/よく	50
	adj_dict - teta	良い → 良かっ/よかっ	50
	adj_renyokei - teta	良く → 良かっ/よかっ	50
Derivational morphology	noun_na_adj + ka	強 → 強化/きょうか	50
	adj + sa	良い → 良さ/よさ	50
	noun + sha	筆 → 筆者/ひっしゃ	50
	noun + kai	茶 → 茶会/ちゃかい	50
	noun_na_adj + kan	同 → 同感/どうかん	50
	noun_na_adj + sei	毒 → 毒性/どくせい	52
	noun_na_adj + ryoku	馬 → 馬力/ばりき	50
	fu + noun_reg	利 → 不利/ふり	50
	dai + noun_na_adj	事 → 大事/だいじ	50
	jidoshi - tadoshi	出る → 出す/だす	50
	Total		1017

Table 5.5: Table 2 from [Als+21a]. Original caption: "List of relations between the words of the Japanese dataset and corresponding number of unique analogies before data augmentation." For details on the meaning of the morphological transformations, see [Kar+18].

Language	# Analogies	# Features with analogies (% of all features)	# Words with analogies (% of vocabulary)
Adyghe	3666973	24 (100.00%)	11155 (100.00%)
Albanian	378591	140 (100.00%)	9666 (100.00%)
Arabic	456689	196 (100.00%)	12942 (100.00%)
Armenian	391054	220 (100.00%)	14415 (100.00%)
Asturian	608932	139 (74.33%)	9122 (99.61%)
Bashkir	3912246	24 (100.00%)	9231 (100.00%)
Basque	43754	1584 (95.77%)	8918 (99.34%)
Belarusian	1025983	56 (100.00%)	8769 (100.00%)
Bengali	163424	58 (100.00%)	3759 (100.00%)
Bulgarian	593920	95 (100.00%)	11290 (100.00%)
Czech	598680	180 (94.24%)	12056 (99.90%)
Danish	7274570	14 (100.00%)	11205 (100.00%)
Dutch	2031211	25 (100.00%)	11181 (100.00%)
English	10006487	5 (100.00%)	16245 (100.00%)
Estonian	641478	108 (100.00%)	10262 (100.00%)
Finnish	508684	197 (100.00%)	18231 (100.00%)
French	1029926	49 (100.00%)	15220 (100.00%)
German	2108502	37 (100.00%)	13174 (100.00%)
Greek	811576	177 (100.00%)	13668 (100.00%)
Hebrew	1095028	54 (100.00%)	8957 (100.00%)
Hindi	246605	211 (100.00%)	8916 (100.00%)
Hungarian	1062552	93 (100.00%)	16747 (100.00%)
Irish	2248336	89 (100.00%)	13169 (100.00%)
Italian	990860	51 (100.00%)	16016 (100.00%)
Kannada	133094	95 (100.00%)	3049 (100.00%)
Kurmanji	2836118	104 (98.11%)	16209 (99.99%)
Latin	447718	151 (100.00%)	16141 (100.00%)
Latvian	984308	80 (100.00%)	14127 (100.00%)
Persian	375639	136 (100.00%)	9323 (100.00%)
Polish	1023982	111 (100.00%)	14779 (100.00%)
Portuguese	668308	76 (100.00%)	12921 (100.00%)
Romanian	945689	59 (100.00%)	12380 (100.00%)
Russian	978081	89 (91.75%)	17227 (99.94%)
Sanskrit	822359	120 (100.00%)	8473 (100.00%)
Slovak	2026778	39 (100.00%)	7442 (100.00%)
Slovene	900301	99 (100.00%)	10189 (100.00%)
Sorani	246077	244 (97.99%)	10158 (99.95%)
Spanish	725601	70 (100.00%)	14445 (100.00%)
Swahili	207967	207 (100.00%)	6419 (100.00%)
Turkish	304609	288 (96.00%)	12650 (99.91%)
Urdu	245343	217 (100.00%)	5192 (100.00%)
Uzbek	7312	84 (100.00%)	936 (100.00%)
Welsh	799086	63 (100.00%)	8820 (100.00%)
Zulu	348500	228 (98.70%)	9613 (99.97%)

Table 5.6: Statistics of high resource languages from Sig19. Languages in bold are also present in Sig16. Note that only a training set is available for high resource languages. Languages in red have less than 50000 analogies.

5.4 Associated code, tools, and dissemination

The Siganalogies dataset is made available publicly via the Dorel platform, and publicized via the `recherche.data.gouv.fr` website[8] and the website of the AT2TA (ANR-22-CE23-0023)[9].

In particular, the Dorel repository contains a backup of the original data from Sig16, Sig19, and JBATS, pre-computed APs in most of the dataset languages, as well as a snapshot of the Python code to extract the APs from the source datasets and manipulate them, described in next paragraph.

The Siganalogies code repository on GitHub[10] contains the latest version of the code to produce and manipulate Siganalogies, in particular:

- the necessary code to encode and decode the words of each language at the character level, for use with, for instance, PyTorch;
- the tools to automatically download of pre-computed preprocessed datasets from Dorel;
- the tools to concatenate the datasets of different languages, which were used for the experiments in Chapter 7;
- some utility functions related to the use of APs, for instance, the data augmentation process (see Section 6.3);
- the data preprocessing code for Siganalogies, which is deterministic: assuming no major change in Python is made, 2 different environments (computer, operating system, *etc.*) will result in the same preprocessed data: same character encoding module, same APs extracted, same vocabulary, *etc.*.

The code repository is meant to used as a sub-repository, and offers a simple, one-function interface for most use-cases. By default, pre-computed preprocessed datasets are fetched from Dorel and save them locally, and if it is not possible, the dataset is automatically built and preprocessed from the source data (Sig16, Sig19, JBATS).

5.4.1 Structure of the data in the source datasets

In practice, Sig16 task 1 data contains four files for each language, as in Example 5.5 Triplets are organized in the $A\,f\,B$ order.

Example 5.5: Example data from German of Sig16

With the example of German, we have:

- `german-task-1-train.txt` containing the training data;
- `german-task-1-dev.txt` containing the development data;
- `german-task-1-test.txt` containing the test data;
- `german-task-1-test-covered.txt` containing the subset of the test data with only morphological transformation seen in training, but in our experiments we do not consider this file.

In `german-task-1-train.txt`, the first three triplets are declensions of the verb "*aalen*":

aalen	pos=V,mood=IND,tense=PRS,per=1,num=PL	aalen
aalen	pos=V,mood=IND,tense=PRS,per=3,num=PL	aalen
aalen	pos=V,mood=IND,tense=PST,aspect=PFV,per=2,num=SG	aaltest

[8] `https://recherche.data.gouv.fr/en/dataset/siganalogies-millions-of-morphological-analogies-in-more-than-80-languages`
[9] `https://at2ta.loria.fr/software/`
[10] `https://github.com/EMarquer/siganalogies`

Sig19 task 1 is organized as folders, one for each pair of high-/low-resource language. Each folder contains four files, corresponding to a training set for the high-resource language, and a training, development, and test sets for the low-resource language. Triplets are organized in the $A\,B\,f$ order, unlike in Sig16.

Example 5.6: Example data from German of Sig19

German appears as a high-resource language in three language pairs: German/Middle-High German (`german--middle-high-german` folder), German/Middle-Low German (`german--middle-low-german` folder), and German/Yiddish (`german--yiddish` folder). All three folders contain the `german-train-high` file, with the following first triplets:

Doppeldecker	Doppeldecker	N;DAT;SG
fassen	fassten	V;IND;PST;1;PL
vergiften	vergifteten	V;IND;PST;3;PL

If we take, for instance, the German/Middle-High German pair, it contains three files in addition to `german-train-high`:

- `middle-high-german-train` containing the training data of Middle-High German;
- `middle-high-german-dev` containing the development data of Middle-High German;
- `middle-high-german-test` containing the test data of Middle-High German.

JBATS is organized slightly differently from Sig16 and Sig19: it contains separate files for each relations, grouped in folders corresponding to the four major categories of the dataset, *i.e.*, one file per morphological transformation for the derivational and inflectional morphology categories (see Table 5.5). For each triplet $\langle A, B, f\rangle$, f is found from the file in which the pair A, B is found. For practical reasons, we transformed this format to follow the $A\,f\,B$ format of Sig16, and gathered the triplets of all relations from the derivational and inflectional morphology categories of JBATS into a single file called `japanese-task-1-train.txt`. See Table 5.7 for examples of the the data in the original file `D01 [noun\_na\_adj + ka].txt` and how it is transformed to fit the Sig16 format.

Ex.	Original data `D01 [noun_na_adj + ka].txt`		Data after transformation `japanese-task-1-train.txt`		
1	活性	活性化/かっせいか	活性	[noun_na_adj + ka]	活性化/かっせいか
2	強	強化/きょうか	強	[noun_na_adj + ka]	強化/きょうか
3	高齢	高齢化/こうれいか	高齢	[noun_na_adj + ka]	高齢化/こうれいか

Table 5.7: First three examples in `japanese-task-1-train.txt`. All three come from the original JBATS file `D01 [noun_na_adj + ka].txt` in folder `2_derivational_morphology`.

5.5 Discussion and perspectives

Along our experiments on Siganalogies, we identified several limitations of the data. Firstly, there are character encoding differences between Sig16 and Sig19, explained in Subsection 5.5.2. Secondly, we found representativeness issues due to an uneven and limited distribution of Part Of Speech (POS) tags and obsolete words, presented in Subsection 5.5.1.

The Siganalogies dataset we developed and refined along our experiments, and many perspectives remain for improvement and further experiments. We elaborate on these in Subsection 5.5.3.

5.5.1 Writing systems and alphabet gap

The way words are written has significant impacts on some properties of the data. For instance, it can change the difficulty of the analogy detection and analogy solving tasks, or reduce the transferability of models from one language to another (see Chapter 8). The data for Arabic is a typical example of how character representation can be challenging.

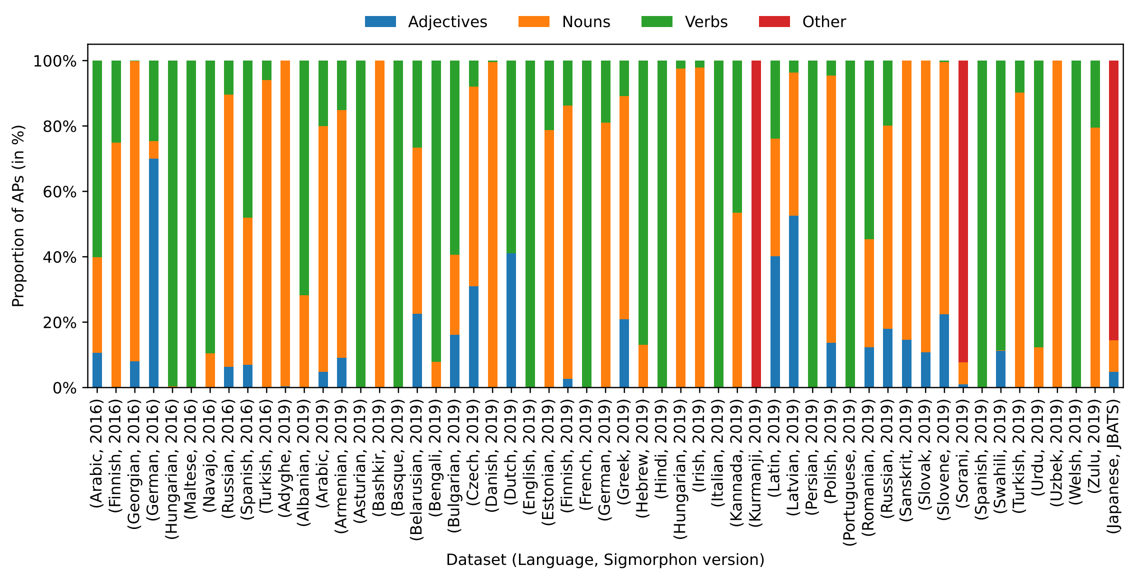


Figure 5.1: Distribution of POS tags among the available APs. "Nouns" and "Adjectives" are cleanly indicated in the features, while "Verbs" regroups multiple verbal forms, including converbs and participles. "Other" contains any POS tag that does not correspond directly to the the other categories, due to feature tags specific to the language.

A first important point is that most languages in Siganalogies do not use the same set of characters. This difference in characters used results in an alphabet gap between some languages, limiting the possibility of transferring morphology, as we show in Subsection 8.1.1 and Section 8.2. As a result, we observe distinct cluster of languages within the high ressource languages of Sig19 in Section 8.2 and Figure 8.7. As can be seen from our experiments in Subsection 8.1.1, this alphabet gap also matters for languages that are present in both Sig16 and Sig19. Notably, a different philosophy is used to represent characters in the two iterations of SIGMORPHON: Sig16 uses romanized writing (see Footnote 4 page 51) for languages such as Arabic, and Sig19 uses the original characters of the language coupled with the Unicode Transformation Format – 8-bit (UTF8) character encoding. This serves as both a limitation and an additional challenge offered by the dataset, that we used to explore some properties of the ANN framework as described in Chapter 8.

Furthermore, the UTF8 character encoding for some languages is a hidden source of additional complexity when using the data. Qualitative analysis of the Arabic data reveals that the UTF8 encoding decomposes each Arabic character into multiple encoded characters, resulting in longer and more complex sequences of characters than expected.

Finally, the data extracted from JBATS contains multiple writings for some words, some using hiragana and katakana, other using kanji. In our experiments, this difference has not been leveraged, however the current implementation of Siganalogies allows for a choice between the two type of forms.

5.5.2 Distribution of Part Of Speech tags and morphological features

For most languages in the Siganalogies dataset, the AP are skewed towards specific POS (or grammatical categories), as can bee seen in Figure 5.1. Additionally, the number of different morphological transformations varies widely from one language to another, as can be seen in Figure 5.2. This can negatively impact the generalization ability of models trained on Siganalogies, for instance for the AE-emb model described in Subsection 7.4.3, as the models only see a subset of the POS of the language.

Additionally, the APs we produce rely on inflectional morphology, as we use data from the task 1 of both Sig16 and Sig19. In this data, each pair of word contain the lemma of the word as well as an inflected form of this lemma. As a consequence, the APs in Siganalogies always contain two lemmas and two inflected forms, *e.g.*, "*(to) run*" : "*running*" :: "*(to) drive*" : "*driving*". This excludes the notion of re-inflexion, where an inflected form is used as the origin of the morphological transfor-

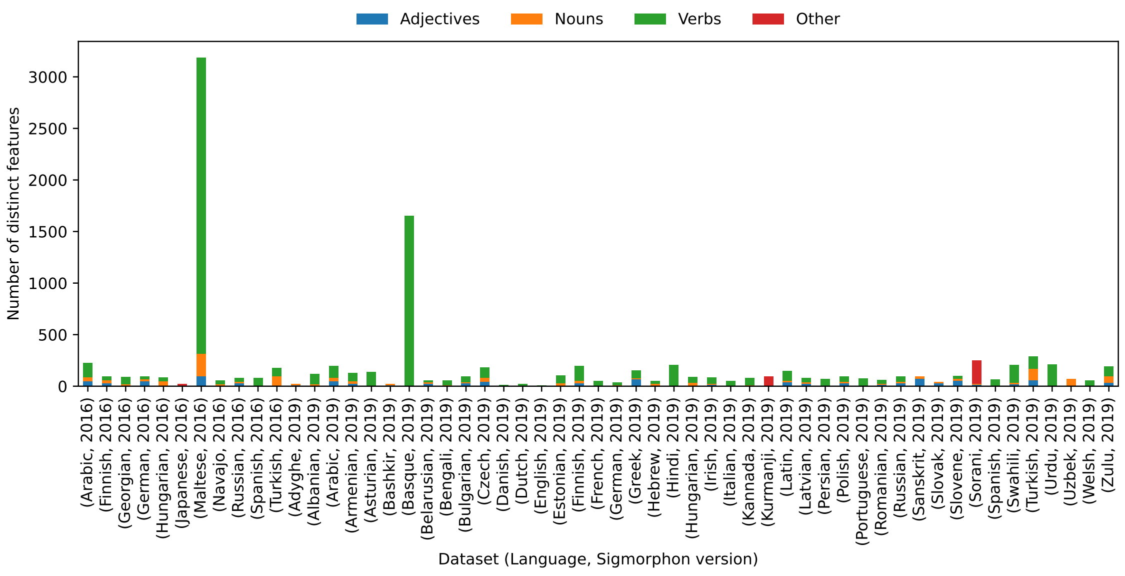


Figure 5.2: Distribution of distinct morphological features per POS tags. "Nouns" and "Adjectives" are cleanly indicated in the features, while "Verbs" regroups multiple verbal forms, including converbs and participles. "Other" contains any POS tag that does not correspond directly to the the other categories, due to feature tags specific to the language.

mation to another inflected form of the same lemma, *e.g.*, "*runs*" : "*running*" :: "*drives*" : "*driving*". In preliminary experiments, we confirmed the possibility of obtaining re-inflexion transformation by matching lemmas between pairs of words, and create corresponding APs. For instance the ratios "*(to) run*" : "*running*" and "*(to) run*" : "*runs*" can be matched to obtain "*runs*" : "*running*". Using re-inflexion data from SIGMORPHON is a reasonable alternative, but the amount of APs that can be obtained must be investigated first.

Anomalous and outdated words. The data in Sig16 and Sig19 contains a number of words that can be considered outdated at best. For instance, the French words in Example 5.7 are rarely used, come from regional dialects, or are outdated according to Wikitionary.

Example 5.7: Outdated and regional words from the French data of Sig19

Below are some words extracted from the French data of Sig19. These words are rarely used, come from regional dialects, or are outdated according to Wikitionary.

- *aoûter*: "to ripen or harden in the August heat", technical, or "to change, regarding weather typical of August" outdated in everyday language;
- *empéguer*: "to glue", Occitan word (regional dialect);
- *estraire*: "to extract", Old French;
- *foler*: "to behave like a crazy person", Old French;
- *numbrer*: "to number, to count", Old French.

The Centre National de Resources Textuelles et Lexicales (CNTRL) offers access to a number of digital and digitalized dictionaries for the French language, covering from Middle French (*Dictionnaire du Moyen Français*[11]) to modern French (9th edition of the *Dictionnaire de l'Académie française*[12]). Among these ressources, *empéguer* and *estraire* do not appear, *foler* and *numbrer* only appear in the *Dictionnaire du Moyen Français*.

[11] http://zeus.atilf.fr/dmf/
[12] https://www.academie-francaise.fr/le-dictionnaire/la-9e-edition

Such outdated words appear in APs built from Sig16 and Sig19, even if they only represent a small part of the dataset. They can be seen as out-of-distribution for the morphology of modern languages, like modern French for our previous examples. Recent versions of the data produced within the SIGMORPHON shared tasks might not share this issue, and could be suitable to complement Siganalogies without having to manually check for out-of-distribution word pairs.

5.5.3 Perspectives for improving Siganalogies

Among other possibilities, it would be of interest to use pairs of inflected forms of the words, instead of pairs of a lemma and an inflected form of this lemma, as mentioned in Section 5.5.

Additionally, thanks to having the full description of morphological features, it would be possible to define more relaxed morphological transformation, such as building APs from pairs with features "Verb;Present;3rd person" and "Verb;Present;1st person" because they share the "Verb;Present" part. Such data would provide intermediate steps in the spectrum between valid and invalid AP, which can help in determining the extent of the dependence of the model on the analogical setting, as we do in Chapter 11 for frame semantics. Other works have considered different granularity of analogy [Ant22; BMP19] and different levels "analogical validity" and their links with human analogical reasoning behavior [MDC17; Mur+20].

Finally, as discussed in Section 8.3, Sig19 offers pairs of more or less related languages that can be used to define cross-lingual APs, and the results on the shared task are available in [McC+19] for comparison with the performance of the analogical approach.

Chapter 6

The ANN framework

Chapter contents

In this chapter, we describe the Analogy Neural Network framework (ANN framework), illustrated in Figure 6.1. The framework can be split into three key elements, namely the embedding model(s), the analogy models, and the data augmentation.

The ANN framework was initially introduced by Lim, Prade, and Richard [LPR21], and was designed for use with pre-trained embeddings to solve semantic analogical equations. In particular, a first rather naive approach to data augmentation for analogical training as well as two NN models to tackle analogy detection and analogy solving were proposed in [LPR21]. These two models, that we call the ANNc and the ANNr, were refined along our experiments. We also integrated several non-parametric approaches into the ANN framework, such as the parallelogram rule [Mik+13] and 3CosMul [LG14] (both defined in Subsection 3.4.1). All these analogy manipulation models on embedding spaces are described in Section 6.2.

The ANN framework is grounded in the axiomatic view of analogy: the architecture of ANNc and ANNr is based on intuitions driven by the postulates described in Section 2.3, and with the data augmentation described in Section 6.4, we train models to fit a given set of postulates by becoming invariant to corresponding permutations. We briefly discuss the components of the ANN framework in Section 6.5.

There are several major differences between the current version of the ANN framework and the version by Lim, Prade, and Richard:

- to obtain suitable representations of words for morphological APs, we use custom morphology oriented embedding models (see Section 6.1) instead of a pre-trained semantics oriented embedding model as done in [LPR21];

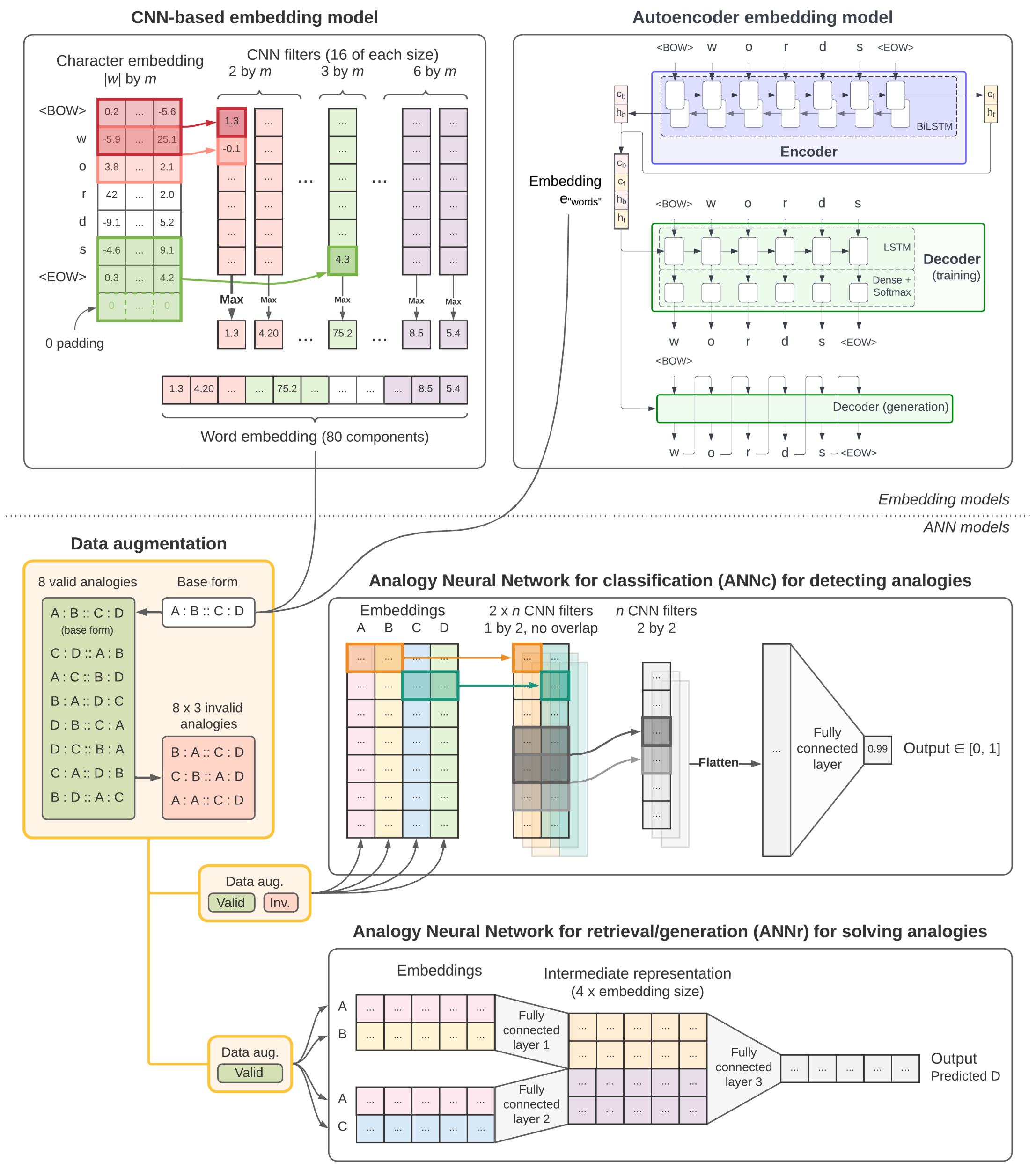


Figure 6.1: Figure 1 from [MC24]. Original caption: “Morphological embedding models, data augmentation, analogy classification (Analogy Neural Network for classification (ANNc)) and analogy retrieval (Analogy Neural Network for retrieval/generation (ANNr)) models.”

- as the embedding model is not pre-trained and was designed to have a small number of parameters, we allow fine-tuning the embedding model during the training of the analogy model (see Section 6.4);
- to reach good performance with non-pre-trained embedding models on analogy solving with ANNr, we use ANNc and analogy detection as a pre-training task (see Subsection 6.4.2);
- because we allow fine-tuning the embedding model during training with ANNr, we need to use more refined training objectives than MSE (see Subsection 6.4.2);
- we improve the data augmentation process by balancing valid and invalid APs, and experiment with different axiomatic settings (see Section 6.3).

As the ANN framework went through multiple iterations along our experiments, we try to clearly specify which experiment uses which version of each component of our approach, both in this chapter and in Chapters 7 and 8.

6.1 Embedding models for morphology

As mentioned in Section 3.3, an important part of DL systems is to design embeddings suitable for the task to accomplish, meaning the embeddings must contain the information necessary to tackle the task.

In our early experiments using pre-trained semantic word embeddings, reproduced in Subsection 7.3.2, we obtained relatively poor performance, which led us to use two different kinds of embedding models. The first one, the CNN-based word embedding (CNN-emb), is inspired from the CNN [LeC+89, see also Subsection 3.2.2] proposed by Vania [Van20]. It is designed specifically for applications on word morphology, and detailed in Subsection 6.1.1. The second, the AE-based word embedding (AE-emb), uses the AE technique [Kra91, see also Subsection 3.3.3] and LSTM [HS96, see also Subsection 3.2.3] as described in Subsection 6.1.2.

Contrary to CNN-emb, this second model is able to generate a word from any embedding. It is inspired by the sequence generation approach by Wang and Lepage [WL20] for analogy solving with English sentences, and by the character level AE from Chollet [Cho17]. In other words, for the AE-emb we have direct access to (an estimate of) the inverse embedding function e^{-1} mentioned in Subsection 3.4.1.

6.1.1 The CNN-based word embedding

The CNN-based word embedding (CNN-emb) is inspired from the work of Vania [Van20]. The embedding models using CNN presented in [Van20, Subsection 2.3.2] are designed for language modelling (see Example 3.2), *i.e.*, to predict the next word in a text. The model combines a CNN and an LSTM: the CNN encodes morphological features from salient sub-words, and the LSTM uses these features to predict the next words. In our work, we use only the CNN part, that we call CNN-emb and describe in this subsection.

Computing the embedding corresponding to a given word. CNN-emb learns to detect key morphological patterns from the characters forming a word. To do so, the input of the model are the characters of a word. First, the characters are embedded into vectors of size m learned together with the rest of the word embedding model. As schematized in Figure 6.2, multiple CNN filters are used, and each filter goes over the character embeddings by spanning over the full embeddings of 2 to 6 characters, resulting in filter sizes between 2 by m and 6 by m. For each filter, the model computes the maximum output to serve as a component of the word embedding. This last operation keeps only salient patterns detected by each of the filters, and forces each CNN filter to specialize in identifying a specific pattern of characters. We use 16 filters of each size between 2 to 6, resulting in embeddings of 80 components. By using character embeddings to encode character features, the model is able to capture patterns of characters, such as "*-ing*", but also patterns based on features of characters, like "*vowel-vowel-consonant*". This flexibility is useful to deal with phenomena like euphony detailed in Section 4.1 (*e.g.*, "*far*" becomes "*further*" when adding the suffix "*-ther*"). These character patterns correspond to morphemes, the minimal units of morphology (see Section 4.1). As the main components of this embedding model are CNN filters, we coin it the CNN-based model.

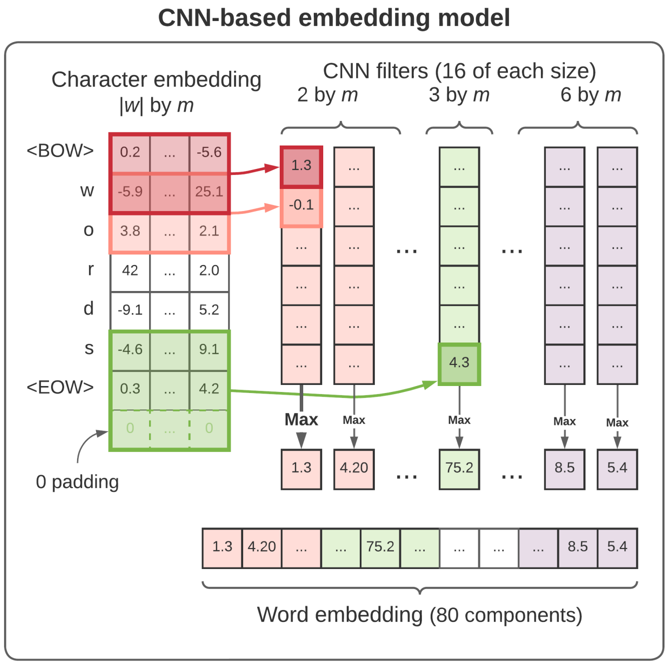

CNN-based embedding model
Character embedding
|w| by m
CNN filters (16 of each size)
2 by m
3 by m
6 by m
<BOW>
w
o
r
d
s
<EOW>
0 padding
Max
Word embedding (80 components)

Figure 6.2: CNN-based word embedding (zoom on Figure 6.1)

Hyperparameters 6.1

To describe CNN-emb, we use the following hyperparameters:

- m: the size of the character embeddings;
- $2, 6$: the smallest and largest filter size used;
- 16: the number of CNN filters of each filter size.

The values of $2, 6$ and 16 are used to compute the size of the word embedding (the hyperparameter n) as $n = (6 - 2) \times 16 = 80$.

Finding the word corresponding to an embedding. When used for analogy solving, the CNN-emb model is unable to produce the word corresponding to an arbitrary embedding. Instead, we need to compute the embeddings of each word in a list of candidates and select the most relevant word according to some similarity measure, *e.g.*, cosine similarity. In experiments from [Als+21c], reported in Subsection 7.4.2, we observed that cosine similarity slightly outperforms Euclidean distance. The performance difference is not significant overall, and we use cosine similarity in our later experiments.

With this method, the space of solution for analogy solving is closed as the candidates form a finite set. This property is not an issue when working with the Closed World Assumption (CWA) (*i.e.*, we know all the solution candidates, and no solution outside of these is accepted), but with the Open World Assumption (OWA), we would theoretically have to compute the embeddings of all possible character strings, which is not feasible.

6.1.2 The AE-based word embedding

Our AE-based word embedding (AE-emb) stems from the above-mentioned limitation of CNN-emb for analogy solving. To tackle this issue, we use an AE model that computes the embedding of a word from its characters and learns to generate words from embeddings.

As mentioned in Subsection 3.3.3, an AE is composed of an encoder e and a decoder e^{-1}. In our setting, the encoder encodes a word w into an embedding $e(w)$, and the decoder the decodes $e(w)$ back into the word $e^{-1}(e(w))\widehat{w}$. To minimize information loss and properly decode the embedding, the model is trained using an reconstruction task: it encodes words and then decode the resulting

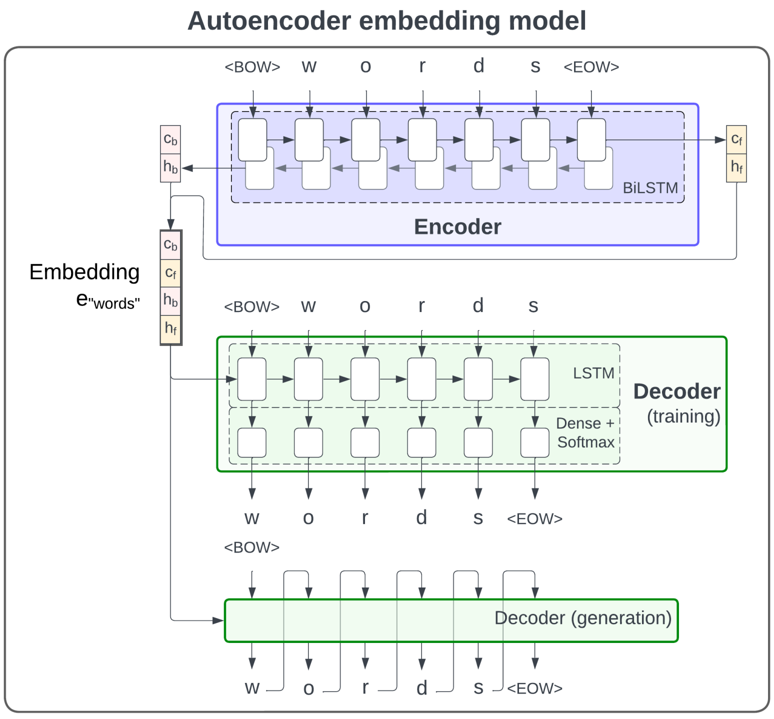


Figure 6.3: AE-based word embedding (zoom on Figure 6.1)

embeddings back into the original words, and the AE is trained to minimize the difference between the original and the decoded words.

The architecture for the model is a character-level sequence-to-sequence AE model, based on the model described in the work of Chollet [Cho17]. Our AE-emb is a character-level model, like the CNN-emb, meaning that we see a word as a sequence of characters. We use BiLSTM [HS96] for the encoder and LSTM [GS05] for the decoder of our AE. These two NNs are designed to handle sequences and store information in a fixed-size memory (see Subsection 3.2.3), which allows us to have embeddings of a constant size no matter the length of the word.

Hyperparameters 6.2

To describe AE-emb, we use the following hyperparameters:

- m: the size of the character one-hot vector, *i.e.*, the number of character recognized by the model;
- $|e(w)|$: the size of the word embedding (must be a multiple of 4).

The value of $|e(w)|$ is used to compute other hyperparameters:

- $|e(w)|/4$: the size of the hidden state of the encoder, with $|e(w)|/4 = |h_f| = |h_b| = |c_f| = |c_b|$;
- $|e(w)|/2$: the size of the hidden state of the decoder, with $|e(w)|/2 = |h| = |c|$.

Computing the embedding corresponding to a given word. Let us now detail, step by step, the process of encoding a word w, which corresponds to the blue part in Figure 6.3.

1. Each character of w is encoded into a one-hot vector (see Subsection 3.3.1).
2. The one-hot encoded characters form are fed into the encoder, which is a BiLSTM. This layer outputs four vectors: the last hidden state h_f and cell state c_f in the forward direction, and similarly h_b and c_b for the backward direction.

3. The concatenation $e(w) = \text{concat}(h_f, h_b, c_f, c_b)$ of these vectors is the embedding of the word.

Finding the word corresponding to an embedding. Decoding a word w is a process similar to encoding it, albeit inverted, and corresponds to the green part in Figure 6.3. We use an LSTM followed by a fully-connected layer with softmax activation function for the decoder. Notice that there are two variants of the decoding process. Usually, generating a sequence with an LSTM is done by a step by step process where each character of the output is predicted one after the other, with the previous character used as input to predict the next one. The first character is primed with beginning of word (BOW) character that marks the start of the word, and prediction is stopped when we encounter an end of word (EOW) character.

Remark 6.1

This iterative generation process is the main reason we use a LSTM and not a BiLSTM for the decoder, as the former is easier to use for iterative generation.

During training, we accelerate training by using teacher forcing, a variant of the decoding process described above where the characters of the word to predict are used in place of the predicted characters as input for the next steps, as illustrated in Figure 6.3. In that process, we know beforehand how many characters to generate. The iterative generation process corresponds to the second and teacher forcing to the first green block in Figure 6.3.

Below, the decoding steps are detailed.

1. The input of the first step of the decoder is the above-mentioned embedding, split into two states $h = \text{concat}(h_f, h_b)$ and $c = \text{concat}(c_f, c_b)$ which initialize the memory of the LSTM.
2. For each character, the previous character encoded as a one-hot vector is fed to the LSTM, which outputs a vector.
3. The output of the LSTM is then transformed into a probability distribution over all the characters available in the dataset. To do so, we use the fully-connected layer mentioned above, such that we have one value for each possible character. Then, the softmax activation function transforms the value
4. We repeat steps 2 and 3 for the other characters in the word, following either the iterative generation process or teacher forcing.

Multiple strategies can be adopted to select the each character in the decoded word, as the model outputs a probability distribution over all possible characters. We can either take the most probable character for each position (*max*), which will result in a deterministic process, or sample characters following the probability distribution (*sample*). The second option is particularly meaningful for the iterative generation process, as the chosen character will impact the next ones, similarly to a butterfly effect. Each generation using *sample* may result in a different generated word. If not specified otherwise, in our experiments we use *max* for generation using AE-emb, to have a deterministic process.

Remark 6.2

If we repeat the iterative generation multiple time using *sample*, words that are frequently generated are very likely to correspond to the embedding. This corresponds to a Monte Carlo estimation of the plausible generation results for a given embedding. This process has been implemented in Analogy Neural Network web application (ANNa), as described in Chapter 9.

Remark 6.3

It is possible to use AE-emb in a retrieval setting by not using the decoder.

6.2 Analogy detection and solving models on embeddings

To manipulate APs, we consider multiple NN models, building upon the embedding models. Here we describe the structure and inner workings of these models, as well as how to use them for analogy detection and analogy solving. We also describe three approaches that do not rely on NNs to solve analogical equations, namely, 3CosAdd [Mik+13], 3CosMul [LG14], and the parallelogram rule.

In this section, we assume that one of the above-mentioned embedding models produced the embeddings $e(A), e(B), e(C)$ of words A, B, C (and $e(D)$ for word D for analogy detection). For analogy solving, it is also necessary to find the word $\widehat{w}$ corresponding to the embedding $\widehat{e(w)}$. This can be done by retrieval (see Subsection 6.1.1) or generation (see Subsection 6.1.2). Note that 3CosAdd and 3CosMul are retrieval methods designed around analogy, and are not suited for generation. The approaches mentioned here can be applied to any embedding model, if the embeddings are always of the same, fixed, size.

6.2.1 Analogy solving with the parallelogram rule, 3CosAdd or 3CosMul

As mentioned in Subsection 2.3.1, the parallelogram rule is one of the interpretations of the axiomatic view of analogy, and has been used for analogy solving since early works on word embeddings [MYZ13; Mik+13], as explained in Subsection 3.4.1.

Two common (equivalent) ways to write the parallelogram rule are:

$$A : B :: C : D \iff e(A) - e(B) = e(C) - e(D), \tag{6.1}$$

$$A : B :: C : D \iff e(D) = e(C) + e(B) - e(A). \tag{6.2}$$

To compute the solution of an analogical equation $A : B :: C : w$, the embeddings $e(A), e(B), e(C)$ are computed by the embedding model and Equation (6.2) is used to compute $\widehat{e(w)} = e(B) - e(A) + e(C)$.

Then, $\widehat{e(w)}$ is used to find the word $\widehat{w}$ solution to the analogical equation. With the AE-emb model, $\widehat{w}$ is obtained using the decoder and with the CNN-emb, retrieval is used instead, following Subsections 6.1.1 and 6.1.2.

Remark 6.4

Equations (6.1) and (6.2) can both be used for analogy detection, by checking if we have equality. When using embeddings, a more flexible approach by checking how far we are from equality is more suitable. To do so, we can apply a relative (*i.e.*, dependant on $e(A), e(B), e(C), e(D)$) or absolute (*i.e.*, constant) threshold to make the decision, for instance, we could compute $|(e(A) - e(B)) - (e(C) - e(D))|$ and draw a conclusion with some threshold t:

$$|(e(A) - e(B)) - (e(C) - e(D))| \leq t \implies A : B :: C : D.$$

3CosAdd [Mik+13] and 3CosMul [LG14] are retrieval approaches to solve analogical equations within an embedding space and are often used even if they have known limitations (see the last paragraph of Subsection 3.3.4). For instance, as retrieval approaches, both methods rely on the embeddings of candidate solutions to solve the equation, and are thus limited by the set of candidates.

In Section 7.4, we report results from [Mar+22a; MC24] on the performance 3CosAdd and 3CosMul applied on the embeddings pre-trained with ANNc to measure the improvement brought by ANNr.

6.2.2 ANNc for analogy detection

The Analogy Neural Network for classification (ANNc) follows the idea that a quadruple A, B, C, D constitutes a valid AP $A : B :: C : D$ if A and B differ in the same way as C and D. If this is true for all the features of A, B, C, D, *i.e.*, for all dimensions of the embeddings $e(A), e(B), e(C), e(D)$, then the AP holds true.

The ANNc model is based two CNN layers, respectively corresponding to the ratio (:) and conformity of ratios (::). Each CNN layer is composed of multiple filters (Subsection 3.2.2). The computation of ANNc is described, step by step, below, and illustrated in Figure 6.4 with annotations below for each step of the process. For all CNN filters, the activation function is the ReLU (see Subsection 3.2.1 Equation (3.4)).

1. The input of the model is $e(A), e(B), e(C), e(D)$, each of size n, stacked into a $n \times 4$ matrix.

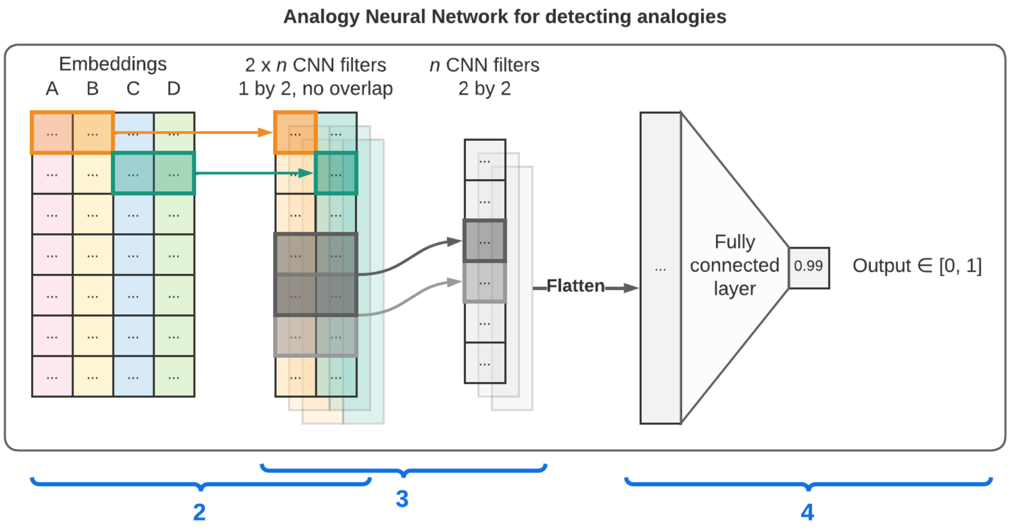


Figure 6.4: Analogy Neural Network for classification (zoom on Figure 6.1)

2. Then, a first set f_1 of CNN filters of size 1×2 is applied on the embeddings, such that for each component $\cdot_i$ of the embedding vector, each filter spans across $e(A)_i$ and $e(B)_i$ simultaneously, and across $e(C)_i$ and $e(D)_i$ simultaneously, with no overlap. Intuitively, this first set of CNN filters extracts the ratios $A : B$ and $C : D$.

3. A second set f_2 of CNN filters of 2×2 is applied on the resulting $|f_1| \times n \times 2$ tensor. Each filter in f_2 moves along the embedding dimension one component at a time, as illustrated in Figure 6.4, resulting in a $|f_2| \times (n-1) \times 1$ tensor. This second set of CNN filters compares $A : B$ and $C : D$, which can be seen as the conformity of ratios.

4. Finally, the output of f_2 is flattened and fed to a fully-connected layer f_3 with a single output (*i.e.*, a perceptron) and sigmoid (see Subsection 3.2.1 Equation (3.5)) as the activation. The sigmoid is defined on $\mathbb{R} \mapsto (0, 1)$, and ensures the output of the fully-connected layer can be interpreted as a classification result: 0 for non-APs, 1 for APs.

While it is possible to assume that embedding components are independent from each other, it is rarely the case in practice. To handle dependent components, filters in f_2 have a size of 2 which results in overlaps between adjacent embedding components, and the fully connected layer mixes the results of all filters from f_2 over all dimensions.

Remark 6.5

The output x of ANNc is the result of a sigmoid and can never take exactly 0 nor 1. To obtain a proper AP/non-AP classification, we use a threshold $t \in [0, 1]$: $x < t$ for non-APs, $x \geq t$ for APs. In our setting, a midpoint threshold $t = 0.5$ gives good results, but a more careful choice could be performed to maximize the performance of the model.

Remark 6.6

The distance between the model output x and the threshold t, *i.e.*, how close x is to either 0 or 1, can be interpreted as the confidence of the model in its prediction. With $t = 0.5$, for the confidence we use $|x - 0.5| * 2$ which gives us a percentage. If we have close to 0% of confidence, the model output is close to 0.5, *i.e.*, the model is not sure in which class to put the quadruple. Conversely, we have close to 100% of confidence if the model output is close to 0 or 1.

Remark 6.7

It is also possible to use ANNc to solve analogical equations, by maximizing the output of the model over possible solutions. We applied this approach in Subsection 7.4.1. This method is most adapted to a retrieval approach, but it is possible to envision generating candidate solutions (for instance, as was done in Alea) or even to maximize the classification score in a machine learning generation algorithm (for example, using a genetic algorithm).

Hyperparameters 6.3

To describe ANNc, we use the following hyperparameters:

- n: the size of the embeddings $e(A), e(B), e(C), e(D)$;
- $|f_1|$: the number of CNN filters in f_1;
- $|f_2|$: the number of CNN filters in f_2.

6.2.3 ANNr for analogy solving

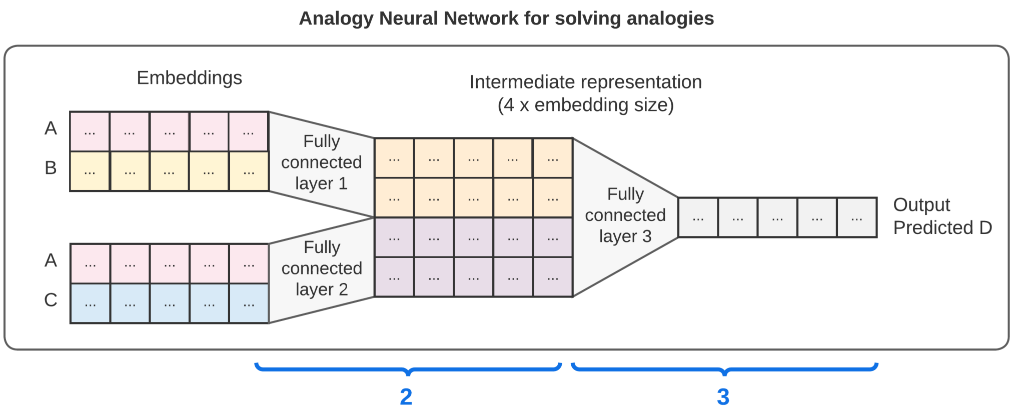


Figure 6.5: Analogy Neural Network for retrieval/generation (zoom on Figure 6.1)

Solving the analogical equation $A : B :: C : w$ can be tackled by finding how $e(B)$ differs from $e(A)$ and generating an $e(w)$ that differs from $e(C)$ in the same way. Exchange of the Means (see Section 2.3) allows us to obtain $A : C :: B : w$, so we can apply the same operation as before, to obtain $e(w)$ from $e(B)$ using the difference between $e(A)$ and $e(C)$. The Analogy Neural Network for retrieval/generation (ANNr) follows this intuition by using a two step process illustrated in Figure 6.5.

Another intuitive explanation is based in the idea of the parallelogram: given a parallelogram $ABCw$ where point w is missing, it is possible to find the location of w by applying the vector $\overrightarrow{AB}$ on C or $\overrightarrow{AC}$ on B.

More specifically, ANNr uses three fully-connected layers in a process described below. Figure 6.5 contains annotations for each step of the process.

1. The input of the model is $e(A), e(B), e(C)$, each of size n.
2. Two separate fully-connected layers f_1, f_2 with ReLU activation are applied respectively on the concatenation of A and B and the concatenation of A and C. Intuitively, f_1 determines the relation between $e(A)$ and $e(B)$ on the one side while keeping the key content of $e(B)$, and similarly for f_2 on $e(A), e(C)$.
3. The outputs of f_1 and f_2 are concatenated and fed into a last fully connected layer f_3, without activation function, which generates $\widehat{e(w)}$ the embedding of the predicted $\widehat{w}$.

The last step is to obtain $\widehat{w}$ from $\widehat{e(w)}$, by using either retrieval (see Subsection 6.1.1) or generation (see Subsection 6.1.2).

Remark 6.8

The intuitions given above rely on Exchange of the Means (implicitly for the parallelogram). However, in some cases Exchange of the Means might be undesirable, as discussed in Subsection 2.3.2. In these settings, the intuition of the model changes slightly: f_1 only determine the relation between $e(A)$ and $e(B)$, and f_2 only extracts the key content from $e(C)$ using $e(A)$ as a form of context.

Remark 6.9

In preliminary experiments, we observed that using different weights for f_1, f_2 achieved better performance than sharing the weights (which would result in a Siamese architecture). This indicates a need for asymmetry between the relations between A and B on one side, and between A and C on the other side. The different weights for f_1, f_2 are also necessary for the intuition described in Remark 6.8.

Hyperparameters 6.4

To describe ANNr, we use the following hyperparameters:

- n: the size of the embeddings $e(A), e(B), e(C), e(D)$;
- whether f_1, f_2 share their parameters or not (see Remark 6.9). By default, parameters of f_1, f_2 are not shared.

6.3 Data augmentation

The postulates of APs described in Section 2.3 and recalled below can be used to generate multiple valid APs (positive examples) and invalid APs (negative examples) from each AP in our dataset. This process is a data augmentation process that extends the amount of data available for training, and is necessary to obtain the negative examples required for training the classifier. Finally, it allows the models to learn how to fit the axiomatic view of analogy by becoming invariant to them.

We consider 2 main ways to obtain positive examples using the postulates of APs:

1. so called trivial APs can be obtained using Reflexivity of Conformity and Identity, as any value for the elements involved in the AP would result in a valid AP;
2. if examples of valid APs are available, it is possible to extend them using the equivalences defined by the postulates of APs.

As for negative examples, we identify 3 systematic ways using the postulates of APs:

3. it is possible to generate invalid APs by breaching constraint postulates; this can be done by altering examples of valid APs as done in the work of Lim, Prade, and Richard [LPR21], but random data could also be used;
4. if examples of valid APs are available and a certain assumption is made as in [LPR21] (we detail the assumption in Appendix C.3.2 Definition 3.2), it is possible to produce negative examples using the permutations that are not equivalent to a valid AP;
5. if examples of invalid APs are available, it is possible to extend them using the equivalences defined by the postulates of APs, in the same way as 2. for positive examples.

The data augmentation process used in the ANN framework went through several iterations as we addressed limitations and experimented with additional features. In most of our work, we follow the approach of Lim, Prade, and Richard [LPR21] and use only 2. for positive and 3. and 4. for negative examples, as described in Subsection 6.3.1 below. In [MMC22], we extend the process as described in Subsection 6.3.2 to account for variants of APs where Exchange of the Means is not accepted, following the reasons discussed in Subsection 2.3.2. A more comprehensive generalization of the data augmentation process is proposed in Appendix C.3, in which more control is offered on the postulates and on the assumptions made on the data. However, this last generalization is not yet supported by experiments and was pushed to the appendices for readability.

6.3.1 Initial version of the data augmentation process

The data augmentation defined in [LPR19; LPR21] relies on the definition of APs as quadruples fulfilling Symmetry of Conformity, Exchange of the Means, Reflexivity of Conformity, and Uniqueness.

Valid APs. By using Symmetry of Conformity and Exchange of the Means, and given a valid AP $A : B :: C : D$, we generate 7 additional APs. For each of the 8 equivalent permutations, we list below one of the possible ways to obtain it:

- $A : B :: C : D$, the base form;
- $A : C :: B : D$ by Exchange of the Means on the base form:
  $A : B :: C : D \rightarrow A : C :: B : D$;
- $B : D :: A : C$ by Symmetry of Conformity on the preceding form:
  $A : B :: C : D \rightarrow A : C :: B : D \rightarrow B : D :: A : C$;
- $B : A :: D : C$ by Exchange of the Means on the preceding form:
  $A : B :: C : D \rightarrow A : C :: B : D \rightarrow B : D :: A : C \rightarrow B : A :: D : C$;
- $D : C :: B : A$ by Symmetry of Conformity on the preceding form:
  $A : B :: C : D \rightarrow A : C :: B : D \rightarrow B : D :: A : C \rightarrow B : A :: D : C \rightarrow D : C :: B : A$;
- $C : D :: A : B$ by Symmetry of Conformity on the base form:
  $A : B :: C : D \rightarrow C : D :: A : B$;
- $C : A :: D : B$ by Exchange of the Means on the preceding form:
  $A : B :: C : D \rightarrow C : D :: A : B \rightarrow C : A :: D : B$;
- $D : B :: C : A$ by Symmetry of Conformity on the preceding form:
  $A : B :: C : D \rightarrow C : D :: A : B \rightarrow C : A :: D : B \rightarrow D : B :: C : A$;

Applying Symmetry of Conformity or Exchange of the Means any number of times on any of this set of 8 forms will produce a form already in the set: we say the set is stable by application of the postulates. As all APs in the set are equivalent, the set can be called an equivalence class. In Example 6.1 below, we list the 8 forms for an example.

Example 6.1: Permutations of a valid AP

"*dog*":"*dogs*"::"*cat*":"*cats*" is a valid AP in morphology. Using Symmetry of Conformity and Exchange of the Means, we obtain a total of 8 valid APs:

- "*dog*":"*dogs*"::"*cat*":"*cats*";
- "*cat*":"*cats*"::"*dog*":"*dogs*";
- "*dog*":"*cat*"::"*dogs*":"*cats*";
- "*cat*":"*dog*"::"*cats*":"*dogs*";
- "*dogs*":"*cats*"::"*dog*":"*cat*";
- "*cats*":"*dogs*"::"*cat*":"*dog*";
- "*dogs*":"*dog*"::"*cats*":"*cat*";
- "*cats*":"*cat*"::"*dogs*":"*dog*".

All of the 8 forms above are valid APs in morphology, even if they appear to focus on different morphological transformations.

Invalid APs. From each of the 8 valid APs, we generate invalid APs following [LPR19; LPR21]. Given a valid AP $A : B :: C : D$, we generate the following invalid APs: $B : A :: C : D$, $C : B :: A : D$, and $A : A :: C : D$, for a total of 24 invalid APs. An example of this process for a morphological AP if detailed in Example 6.2 below.

To justify the generation of invalid APs, consider that a typical non-trivial AP $A : B :: C : D$ is such that $C \neq D$. In such a situation, writing $A : A :: C : D$ contradicts Reflexivity of Conformity, Exchange of the Means, and Uniqueness: Reflexivity of Conformity which states that if $A : B :: A : B$, and by Exchange of the Means $A : A :: B : B$ are valid APs for all A, B. Then by Uniqueness, as $A : A :: B : B$, $A : A :: B : C \implies C = B$. The latter statement is also known as Strong Identity, *i.e.*, Uniqueness applied on Identity.

As for $B : A :: C : D$ and $C : B :: A : D$, we have to look at all possible permutations of $\langle A, B, C, D\rangle$ for the explanation. As stated in [LPR21], the $4! = 24$ permutations of $\langle A, B, C, D\rangle$ can be partitioned in 3 non-overlapping subsets, and each one is a class of equivalence by application of Symmetry of Conformity and Exchange of the Means. The 3 permutations $\langle A, B, C, D\rangle$, $\langle B, A, C, D\rangle$, and $\langle C, B, A, D\rangle$ are taken from each of the 3 equivalence classes, $B : A :: C : D$ and $C : B :: A : D$ allow to reach the classes of equivalence in which $A : B :: C : D$ is not present. Considering that $B : A :: C : D$ and $C : B :: A : D$ are invalid APs if $A : B :: C : D$ is a valid APs does not follow from the postulates and is a strong assumption. Nonetheless, when considering practical examples such as Example 6.2 below, the assumption appears sensible for a rather strict view of APs.

Example 6.2: Invalid permutations of a valid AP

"*dog*":"*dogs*"::"*cat*":"*cats*" is a valid AP in morphology. Using the permutations $B : A :: C : D$, $C : B :: A : D$, and $A : A :: C : D$, we obtain the following.

- "*dogs*":"*dog*"::"*cat*":"*cats*" corresponds to $B : A :: C : D$. For "*dogs*":"*dog*" we remove the suffix "*-s*", while for "*cat*":"*cats*" the suffix is added, so the AP is considered invalid.
- "*cat*":"*dogs*"::"*dog*":"*cats*" corresponds to $C : B :: A : D$. The morphological transformation underlying "*cat*":"*dogs*", which can be interpreted as "replacing "*cat*" by "*dogs*"", is not the same as in "*dog*":"*cats*", and the AP is considered invalid.
- Finally, "*dog*":"*dog*"::"*cat*":"*cats*" corresponds to $A : A :: C : D$, and goes against Strong Identity: "*dog*" is "*dog*", but "*cat*" is different from "*cats*".

Data augmentation after applying the embedding model. In our setting, the embedding of a word will be the same no matter its position in the AP. Therefore, from our first work using the data augmentation [Als+21a] on morphological APs, we apply data augmentation after the embedding model. That way, an embedding computed once is reused for multiple permutations, instead of re-computing it for each permutation of the same elements. Note that in Chapter 11, a similar data augmentation process is used, but as the BERT embeddings we use differs for different permutations of the elements, data augmentation is performed before computing the embedding.

6.3.2 Variants of the data augmentation process when Exchange of the Means is not accepted

In [MMC22], and as reported in Section 7.6, we experiment with Exchange of the Means among the most discussed postulate [Ant22, see also Subsection 2.3.2]. We consider three settings, described below, that correspond to accepting (EM), explicitly rejecting ($\overline{EM}$), and not taking into account ($\neg EM$) the Exchange of the Means. As a basis, we consider the full set of postulates accepted in [LPR21], which includes: Symmetry of Conformity, Inversion of Ratio, Exchange of the Means, Identity, Reflexivity of Conformity, as well as Uniqueness and its implications of Strong Identity and Strong Reflexivity of Conformity. When removing the Exchange of the Means, some permutations derived from it in Subsection 6.3.1 can instead be found with the Inversion of Ratio.

Setting EM corresponds to the standard setting where Exchange of the Means is accepted. To differentiate with the other two settings, we call P^{+}_{EM} the set of valid and P^{-}_{EM} the set of invalid

APs in that setting, defined in Subsection 6.3.1 and in Equations (6.3) and (6.4):

$$\begin{aligned} P^+_{EM} = \{&\langle A,B,C,D\rangle, \langle C,D,A,B\rangle, \langle B,A,D,C\rangle, \langle D,C,B,A\rangle, \\ &\langle A,C,B,D\rangle, \langle C,A,D,B\rangle, \langle B,D,A,C\rangle, \langle D,B,C,A\rangle\}, \end{aligned} \tag{6.3}$$

$$P^-_{EM} = \bigcup\nolimits_{\langle A',B',C',D'\rangle \in P^+_{EM}} \{\langle A',A',C',D'\rangle, \langle B',A',C',D'\rangle, \langle C',B',A',D'\rangle\}. \tag{6.4}$$

In setting $\neg EM$, Exchange of the Means is discarded while maintaining Inversion of Ratio, which is the simplest way to refute the Exchange of the Means postulate. In practice, we simply remove permutations that require Exchange of the Means (or Exchange of the Extremes) to be obtained from $\langle A,B,C,D\rangle$. We also remove $\langle C',B',A',D'\rangle$ from P^-_{EM} to obtain $P^-_{\neg EM}$, as its form is related to Exchange of the Means. It results in the following sets of permutations:

$$P^+_{\neg EM} = \{\langle A,B,C,D\rangle, \langle C,D,A,B\rangle, \langle B,A,D,C\rangle, \langle D,C,B,A\rangle\}, \tag{6.5}$$

$$P^-_{\neg EM} = \bigcup\nolimits_{\langle A',B',C',D'\rangle \in P^+_{\neg EM}} \{\langle A',A',C',D'\rangle, \langle B',A',C',D'\rangle\}. \tag{6.6}$$

For $\overline{EM}$ we go one step further in rejecting the Exchange of the Means by considering permutations obtained through Exchange of the Means invalid APs. We add $\langle A',C',B',D'\rangle$ to P^-_{EM} to obtain $P^-_{\overline{EM}}$, as this quadruple permutation is the result of Exchange of the Means. For the valid APs, we use the same permutations as for $\neg EM$:

$$P^+_{\overline{EM}} = P^+_{\neg EM} \tag{6.7}$$

$$P^-_{\overline{EM}} = \bigcup\nolimits_{\langle A',B',C',D'\rangle \in P^+_{\overline{EM}}} \{\langle A',A',C',D'\rangle, \langle B',A',C',D'\rangle, \langle C',B',D',A'\rangle, \langle A',C',B',D'\rangle\} \tag{6.8}$$

As discussed in Appendix D, after we designed the experiments from [MMC22], we identified more relevant ways to generate the invalid permutations to use.

6.3.3 Balancing valid and invalid Analogical Proportions

The initial imbalanced valid and invalid APs. Initially, as described in [Als+21a; Als+21b], we used the full set of valid APs and the invalid APs obtained from the base for training the ANNc models, and the full set of 24 invalid APs was only used during testing. This resulted in 8 valid for 3 invalid APs during training. In the experiments in Subsection 7.3.3, originally reported in [Als+21c, Appendix B], we identified that an imbalance between the number of valid and the invalid APs during training resulted in a corresponding imbalance in the performance on the two classes of APs.

Sampling and up-sampling APs. Accordingly, in [MC24; MMC22], we added sampling to ensure the same amount of valid and invalid training APs. In most experiments, we use the standard setting (see Subsection 6.3.1, also EM in Subsection 6.3.2) with 8 valid APs, and sample a subset of 8 permutations among valid APs. When using the $\overline{EM}$ and $\neg EM$ (described in Subsection 6.3.2) the number of valid and invalid APs differs, and sampling and up-sampling are performed to obtain 8 valid and 8 invalid APs:

- if $n = 8$ permutations are available, all are used;
- if $n > 8$ permutations are available, 8 distinct permutations are randomly selected;
- if $n < 8$ permutations are available, $8 - n$ randomly selected permutations are added, which ensures that each permutation appears at least once. This last case is a form of *up-sampling*.

Filtering out invalid APs with the form of valid ones. Additionally, in [MC24], filtering is added to ensure that the negative APs generated are not in the class of equivalence of the original AP. Let us consider the AP

("*build*", past tense) : ("*build*", past participle) :: ("*go*", past tense) : ("*go*", past participle),

indicating the lemma and the corresponding inflexion. The corresponding words are

"*built*" : "*built*" :: "*went*" : "*gone*", which appears to breach Strong Identity (and after applying Symmetry of Conformity breaches Strong Reflexivity of Conformity). Without filtering, this valid AP would also appear as an invalid AP, which may cause issues, and such cases were reported in [Als+21c].

To summarize, for every AP in the dataset:

1. we compute the full set of valid (P^+) and invalid (P^-) APs;
2. we remove from P^- permutations that have the same form as any of the valid APs: ${P^-}' = P^- \setminus P^+$;
3. we up-sample or sample from ${P^-}', P^+$ to obtain 8 permutations from each.

6.4 Training the models

In this section, we detail the technical aspects of training ANNc and ANNr in Subsections 6.4.1 and 6.4.2, as well as the pre-training required for suitable performance of CNN-emb and AE-emb in Subsections 6.4.3 and 6.4.4. For the most part, this section follows [MC24].

The experiments from [Mar+22a] (reported in Subsection 7.4.4) showed that pre-training the embedding model helps the performance of ANNr. Using pre-training and fine-tuning (see Subsection 3.3.2) allows to first bring the embedding model to a globally viable state before specializing it on the task, in our case analogy solving with ANNr. The pre-training process when using CNN-emb is detailed in Subsection 6.4.3, and in Subsection 6.4.4 for AE-emb.

6.4.1 Training ANNc for analogy detection

Positive and negative examples. As its name indicates and as explained in Subsection 6.2.2, ANNc was designed as a binary classifier. To properly train a binary classifier for analogy detection, we need not only APs but also quadruples of elements that do not form an analogy, so-called invalid APs. In terms of binary classification, valid APs are positive examples, while invalid APs are negative examples. To obtain them, we use the data augmentation process described in Section 6.3. As we progressed in our experiments and addressed limitations, the training process of ANNc evolved with regards to data augmentation and class sampling, as described in Section 6.3.

Training criterion. To train ANNc we use the Binary Cross-Entropy loss, which is a standard for binary classification. Using the class labels 1 for valid and 0 for invalid APs, the Binary Cross-Entropy is written as follows:

$$\begin{aligned}\mathcal{L}_{ANNc}(y, y_{pred}) &= -y\log(y_{pred}) - (1-y)\log(1-y_{pred}) \\ &= \begin{cases} -\log(y_{pred}) & \text{if } y = 1, \\ -\log(1-y_{pred}) & \text{otherwise,} \end{cases}\end{aligned} \tag{6.9}$$

where y is the true class and y_{pred} is the classifier prediction. In such a setting, the classifier outputs a probability y_{pred} to be in class 1, *i.e.*, the probability that the AP is valid. The probability that the AP is invalid, *i.e.*, in class 0, is $1 - y_{pred}$.

6.4.2 Training ANNr for analogy solving

In this subsection, we summarize the training procedure for ANNr. After our initial experiments in [Als+21c], several variants of the training procedure were tested in [Mar+22a]. These variants differ mainly in terms of pre-training of the embedding model and in the criterion used for training ANNr. All the variants are detailed in the paragraph corresponding to the training criterion.

Examples of analogical equations. To train ANNr, we use analogical equations of the form $A : B :: C : x, x = D$. To maximize the number of training examples and improve the adequation between the ANNr and the postulates, we perform data augmentation as described in Subsection 6.3.1. We generate 8 analogical equations from each AP $A : B :: C : D$ in the data: $A : B :: C : x, x = D$, $A : C :: B : x, x = D$, $D : B :: C : x, x = A$, $C : A :: D : x, x = B$, $C : D :: A : x, x = B$, $B : A :: D : x, x = C$, $D : C :: B : x, x = A$, $B : D :: A : x, x = C$.

Remark 6.10

We perform no experiment with the data augmentation variants mentioned in Subsection 6.3.2, so whenever data augmentation is mentioned for ANNr, we refer to the 8 analogical equations mentioned above.

Training criterion. A standard way to train a regression model in ML is to minimize a distance between the predicted and expected points (respectively $\widehat{e(D)}$ and $e(D)$ in our setting), for instance using the L^2 distance or MSE:

$$MSE\left(e(D), \widehat{e(D)}\right) = \frac{1}{n} \sum_{i \in [1,n]} \left(e(D)_i - \widehat{e(D)}_i\right)^2, \tag{6.10}$$

where $e(D)_i$ is the i-th component of $e(D)$ and n is the number of dimensions of the embedding $e(D)$. However, when we trained ANNr together with CNN-emb using MSE in our preliminary experiments, we observed a collapse of the embedding space, defined as "the [embedding model] produce[ing] constant or non-informative vectors" in [BPL22]. For instance, all the embeddings may be moved in a smaller area of the embedding space by multiplying them by 10^{-5}, which minimizes the MSE properly, however the actual performance of the model stays the same as the relative distance between embeddings does not change.

In [Mar+22a], we experimented with several training criteria to mitigate this issue, namely $\mathcal{L}_{\text{norm. other}}$, $\mathcal{L}_{\text{norm. random}}$, $\mathcal{L}_{\text{CEL}}$, and $\mathcal{L}_{\text{all}}$, defined hereafter. In the experiments from [Als+21c; Mar+22a] reported in Subsection 7.4.4, we compare the performance obtained with each of these criteria.

A straightforward way to mitigate the embedding space collapse is to normalize Equation (6.10), the distance between the actual and the predicted embeddings, using the other embeddings in the dataset. Normalizing using the full distribution of distances between embeddings would be very accurate, but computing embeddings on the full dataset for each training step would be computationally heavy.

Instead, in [Als+21c] we introduced $\mathcal{L}_{\text{norm. other}}$, which uses only $e(A), e(B), e(C), e(D)$ the embeddings of A, B, C, D, as they have already been computed before data augmentation. It corresponds to $MSE(e(D), \widehat{e(D)})$ normalized by the average of the pairwise distances between $e(A), e(B), e(C), e(D)$:

$$\mathcal{L}_{\text{norm. other}} = \frac{1 + 6 \times MSE\left(e(D), \widehat{e(D)}\right)}{1 + \sum_{a,b \in \{(A,B),(A,C),(A,D),(B,C),(B,D),(C,D)\}} MSE(e(a), e(b))}, \tag{6.11}$$

with the 1 in the denominator as a safeguard against divisions by 0. This criterion ensures that the distance between $e(D)$ and $\widehat{e(D)}$ is minimized relatively to the distance between the other elements, and relies on the embeddings of A, B, C, D that have already been computed in our training procedure for normalization.

An alternative introduced in [Mar+22a] is to normalize with regards to some random embedding $e(z)$ taken from the dataset:

$$\mathcal{L}_{\text{norm. random}} = \frac{1 + MSE\left(e(D), \widehat{e(D)}\right)}{1 + MSE\left(e(z), \widehat{e(D)}\right)} \tag{6.12}$$

As the model is trained with batches of randomly selected samples, an $e(z)$ unrelated to $e(D)$ can be obtained by permuting $e(D)$ along the batch dimension. This is cheaper to perform than $\mathcal{L}_{\text{norm. other}}$.

We also included in the study [Mar+22a] the Cosine Embedding Loss (CEL), a criterion widely used to minimize distances between embeddings when the angle between the embeddings is more relevant than the euclidean distance:

$$\text{CEL}\left(e(D), \widehat{e(D)}, y\right) = \begin{cases} 1 - \cos\left(e(D), \widehat{e(D)}\right), & \text{if } y = 1 \\ \max(0, \cos\left(e(D), \widehat{e(D)}\right)), & \text{if } y = 0 \end{cases}, \tag{6.13}$$

with $y = 1$ if $e(D)$ and $\widehat{e(D)}$ should be close and $y = 0$ otherwise. For similar reasons as for $\mathcal{L}_{\text{norm. other}}$, given an analogy $A : B :: C : D$ we want $\widehat{e(D)}$ to be closer to $e(D)$ than to $e(A), e(B), e(C)$. When expressed by the CEL, the previous statement becomes:

$$\begin{aligned}\mathcal{L}_{\text{CEL}} = \text{CEL}\left(e(A), \widehat{e(D)}, 0\right) + \text{CEL}\left(e(B), \widehat{e(D)}, 0\right) \\ + \text{CEL}\left(e(C), \widehat{e(D)}, 0\right) + \text{CEL}\left(e(D), \widehat{e(D)}, 1\right).\end{aligned} \tag{6.14}$$

We also considered the sum of all the above loss terms to determine whether their combination overcomes the limitations of each:

$$\mathcal{L}_{\text{all}} = \mathcal{L}_{\text{CEL}} + \mathcal{L}_{\text{norm. other}} + \mathcal{L}_{\text{norm. random}} \tag{6.15}$$

Remark 6.11

Note that in our implementation, the gradient of the loss is propagated to all the involved embeddings, for instance, $e(A), e(B), e(C), e(D), \widehat{e(D)}$ for $\mathcal{L}_{\text{CEL}}$ and $\mathcal{L}_{\text{norm. other}}$. This allows to adapt $e(A), e(B), e(C)$ to ensure they contain the necessary information for analogy solving, but is also the source of the embedding space collapse.

6.4.3 Pre-training CNN-emb for ANNr

In [Als+21c; MMC22], we confirmed the benefits of pre-training CNN-emb before analogy solving with ANNr. The corresponding experiments are reported in Subsection 7.4.4.

Contrary to AE-emb which can be pre-trained as described in Subsection 6.4.4, CNN-emb is not designed to be trained without a task. We leverage the analogical data, which is already needed to train ANNr, to pre-train CNN-emb using ANNc on the analogy detection task. To do so, we follow the standard ANNc training (Subsection 6.4.1), during which the weights of the CNN-emb model are updated starting from a random initialization.

Fine-tuning CNN-emb when training ANNr. When training ANNr combined with CNN-emb (CNN+ANNr), the parameters of the CNN-emb are frozen (*i.e.*, not updated) at the start of training. This allows ANNr to first learn to use the existing embedding space before fine-tuning the latter for analogy solving, and in preliminary experiments, this provided significantly better results than fine-tuning CNN-emb from the start.

Initially, in [Als+21c] CNN-emb was frozen for 10 epochs. Among the results reported in Chapter 7, only Section 7.5 is in this setting. In our other experiments CNN-emb is frozen until ANNr converges, *i.e.*, until there is no improvement in the loss on the development set. This last setting was found to give better results in [Mar+22a].

6.4.4 Pre-training AE-emb for ANNr

This subsection summarizes the task and criterion for pre-training AE-emb, following [Cha+22].

Reconstruction task. AEs are designed to be trained on an reconstruction task: in our setting, a training dataset containing words is presented to the AE-embs which is trained to encode-decode each word and minimize the error between the predicted word and the original one, as mentioned in Subsection 3.3.3. Reconstruction tasks are interesting because they allow to pre-train embedding models without needing labeled data.

Training criterion. We use the Cross Entropy loss to train the AE-emb, as it is the common way to train a model where each output is one among a set of classes: in our case, each output is one among a set of characters. In the following formula, we write $\mathcal{L}_{AE}$ the loss used for AE-emb, which is exactly the Cross Entropy but re-written and simplified for our setting:

$$\mathcal{L}_{AE}(w, decoder(\widehat{e(w)})) = -\frac{1}{|w|}\sum_{i=1}^{|w|}\sum_{c\in V} w_{c,i}\log\left(decoder(e(w))_{c,i}\right), \tag{6.16}$$

where w is the word to encode-decode, $|w|$ is the number of characters to predict including the EOW character, V is the character vocabulary (including the EOW character), $w_{c,i}$ is 1 if the i-th character of w is c and 0 otherwise, $decoder(\widehat{e(w)})$ is the output of the decoder and $decoder(\widehat{e(w)})_{c,i}$ is the predicted probability that the i-th character is c.

Fine-tuning AE-emb when training ANNr. When training ANNr combined with AE-emb (AE+ANNr), we leverage the differences between the expected words and the prediction of the decoder by adding the reconstruction loss term $\mathcal{L}_{\text{AE}}$ of the AE-emb to the $\mathcal{L}_{\text{norm. random}}$ of ANNr. This results in a convex combination of $\mathcal{L}_{\text{norm. random}}$ and $\mathcal{L}_{\text{AE}}$:

$$\mathcal{L}_{\text{AE + ANNr}}(e(D), \widehat{e(D)}) = (1-\lambda)\mathcal{L}_{\text{norm. random}}(e(D), \widehat{e(D)}) + \lambda\mathcal{L}_{\text{AE}}(D, \text{decoder}(\widehat{e(D)})). \quad (6.17)$$

The parameter $\lambda = \min\left(\max\left(epoch/5, 0.01\right), 0.99\right)$ evolves during training, such that at the beginning of the training $\lambda = 0.99$ and ANNr is training almost alone without altering the performance of the AE-emb. After 5 epochs, $\lambda = 0.01$ and $\mathcal{L}_{AE}$ is used almost alone, and the AE-emb is fine-tuned for analogy solving while training ANNr. This process is inspired by results from [Mar+22a] detailed in Subsection 6.4.3, where delaying the fine-tuning of CNN-emb when training ANNr resulted in better performance than fine-tuning CNN-emb from the start. Additionally, in preliminary experiments, using only $\mathcal{L}_{AE}(D, \text{decoder}(\widehat{e(D)}))$ to train AE+ANNr achieved poor performance, which motivated the use of the combined loss.

6.5 Discussion and perspectives

Performance of the ANN framework and design choices. The ANN framework we defined in this chapter offers multiple ways to tackle analogy detection and analogy solving. We detail in next chapter experiments that demonstrate the performance of the ANN framework on analogy detection and analogy solving, and we discuss the main pros and cons of each method in Subsection 7.7.4. In Chapter 7, we also present results from preliminary experiments that guided the design decisions mentioned in the present chapter. These experiments are discussed in more detail in Subsection 7.7.2.

Alternatives to the current models. The ANN framework we study offers multiple ways to tackle analogy detection and analogy solving, and we discuss the main pros and cons of each method in Subsection 7.7.4. Despite the alternatives we propose, it can be argued that the architecture of ANNc and ANNr are not the most fitting for the manipulation of APs, yet they are relatively easy to grasp intuitively and achieve good performance.

An alternative to ANNr was proposed by Mao and Lepage [ML23] for analogy solving between sentences, with different intuitions with regards to the properties of APs. It would be interesting to extend our results to this new architecture.

For AE+ANNr and parallelogram rule combined with AE-emb (AE+par.), it is possible to estimate the likelihood of multiple solutions instead of considering only the most likely characters for each position. To do so, we can use a similar approach as Alea and perform a Monte Carlo estimate of the likely answers. We implemented this system in the ANNa platform (see Chapter 9), which allows to measure the confidence of the model in specific solutions. On some manually crafted examples, the confidence in the invalid answer was low and the expected output had a close confidence, a behavior similar to what we observed in the top 10 candidates with CNN+ANNr.

One limitation of the ANNc and ANNr combination is that multiple models are required to cover analogy detection and analogy solving. We attempted to circumvent this limitation by using ANNc combined with CNN-emb (CNN+ANNc) as a retrieval model, with encouraging results. However, it would be useful to have an approach that is, by design, able to handle both aspects of analogical reasoning. We performed preliminary experiments involving backpropagation through CNN+ANNc to alter specific dimension of the input embeddings. This kind of approach would allow to generate both solutions, by maximizing the classification score of CNN+ANNc, and counterfactual examples (*i.e.*, plausible counterexamples that are close to the decision border [KS20; Kus+17]), by minimizing the classification score. Our preliminary experiments yielded promising results on some artificial data distributions, but many limitations remain to be addressed.

Chapter 7

Quantitative and qualitative analyses of the ANN framework

Chapter contents

This chapter details our experiments with the ANN framework (see Chapter 6) on the Siganalogies dataset (see Chapter 5). For each experiment, we interpret the results and conclude in the corresponding section or subsection. We also conclude this chapter with a discussion of our contributions and results presented throughout Sections 7.3 to 7.5, in Section 7.7.

Before describing our experiments, we give some general information on our experimental setup in Section 7.1, and we summarize technical aspects of the symbolic baselines in **??**. We then detail several experiments on models trained on each language for analogy detection in Section 7.3, and analogy solving in Section 7.4. In Section 7.5, we perform an ablation study on both the analogy detection and analogy solving tasks to determine their tolerance *w.r.t.* perturbations in the embedding space. Experiments with models transferred between languages or trained on multiple languages are described in Chapter 8.

We also study how the analogical data augmentation impacts the behavior of the model for analogy detection when Exchange of the Means is considered differently, following discussions on how to handle postulates of APs that are not suitable for some application settings [Als+22b; Ant22, see also Subsection 2.3.2]. These experiments are described in Section 7.6, and refer to variants of the data augmentation process defined in Subsection 6.3.2.

7.1 General information on the experiments of this chapter

In this section, we explain and define elements of the experimental setup that are common to all our experiments. In Subsection 7.1.1, we define the measures we use to evaluate the performance of our models on different tasks, and list the models themselves and their hyperparameters in Subsection 7.1.2. The manner in which data is split into the training set, development set, and test set is explained in Subsection 7.1.3.

7.1.1 Performance measurement

This subsection contains the definitions of all the measures we use to compare models in our experiments.

Performance on the analogy detection task. To measure the analogy solving performance of a model we use the accuracy. We separate the accuracy for each class, namely, valid or invalid APs. In some cases we also consider the "base" form of the AP, which is the permutation present in the data. In general, classification accuracy is defined as:

Definition 7.1: Accuracy for a class in the analogy detection task

Accuracy for a given class is the rate of examples (in our case, APs) on a dataset for which the expected class is given (in our case, either valid or invalid):

$$\text{acc. on valid APs} = \frac{\#\text{ correctly predicted valid APs}}{\#\text{ valid APs}},$$
$$\text{acc. on invalid APs} = \frac{\#\text{ correctly predicted invalid APs}}{\#\text{ invalid APs}}.$$

When differentiating between the performance on valid and invalid permutations is unnecessary, we aggregate them using the balanced accuracy:

Definition 7.2: Balanced accuracy for the analogy detection task

We call balanced accuracy in the context of the analogy detection task, the arithmetic mean of the accuracy on valid and on invalid APs, without considering the number of examples in each class:

$$\text{bal. acc.} = \frac{\text{acc. on valid APs} + \text{acc. on invalid APs}}{2}.$$

Performance on the analogy solving task. To measure the analogy solving performance of a model we use either the retrieval accuracy for retrieval models or the generation accuracy for the generation models:

Definition 7.3: Accuracy for the analogy solving task

Accuracy for analogy solving is the rate of queries (in our case, analogical equations) on a dataset for which the expected solution is given:

$$\text{acc.} = \frac{\#\text{ correctly predicted solutions}}{\#\text{ analogical equations}}.$$

For retrieval models that provide a ranking of possible solutions, this measure can be extended to consider the top k retrieved candidates:

Definition 7.4: Hit rate at k for the analogy solving task

For hit rate at k (written hit@k), a query (in our case an analogical equation) is considered a success if the expected solution is within the top k retrieved solutions. Hit rate at k for a retrieval model is the rate of such successes on a dataset:

$$\text{hit@}k = \frac{\#\text{ times the expected solution is in the top } k \text{ retrieval results}}{\#\text{ analogical equations}}.$$

By definition, for a given model and dataset, if $k < k'$ then the hit rate at k is lower or equal to the hit rate at k'. Note that hit rate at 1 is exactly the retrieval accuracy.

Performance on the reconstruction task. For the reconstruction task (see Subsections 3.3.3 and 6.4.4), we measure the performance of the AE-emb using word-level accuracy:

Definition 7.5: Word-level accuracy for the reconstruction task

The word-level accuracy for the word reconstruction task is the rate of words where all the characters are correctly predicted, at the expected position:

$$\text{acc.} = \frac{\#\text{ fully correct predicted solutions}}{\#\text{ analogical equations}}.$$

Remark 7.1

The name of word-level accuracy is by opposition with character-level accuracy, which is defined as either: *(i)* the rate of characters correctly predicted over the whole set of predictions, or *(ii)* the average over all predictions of the rate of correctly predicted characters. The first variant gives the same weight to all characters not matter the word, while the second one considers characters in a short word comparatively more important than characters in a longer word. The second variant normalizes the character accuracy by the length of the word containing the character.

Characters shared between two languages. In some of our experiments (see Subsection 7.3.4 and Chapter 8), as we use character-level embedding models, we are interested in comparing the sets of characters present in different settings. The settings can represent the same language (as in Subsection 7.3.4) or different languages (as in Chapter 8). We use the coverage of characters from one set by the characters of another set to compare the sets of characters of the settings:

Definition 7.6: Character coverage

Given two sets of characters $\mathcal{C}_1, \mathcal{C}_2$, the coverage of $\mathcal{C}_2$ by $\mathcal{C}_1$ is:

$$\text{char. coverage} = \frac{|\mathcal{C}_2 \cap \mathcal{C}_1|}{|\mathcal{C}_2|}.$$

Character coverage is not symmetric, so when we need a symmetric measure of the amount of character shared by two sets of characters, we use the Jaccard index of character sets instead:

Definition 7.7: Character Jaccard index

Given two non-empty sets of characters $\mathcal{C}_1, \mathcal{C}_2$, the Jaccard index is defined as:

$$\text{char. Jaccard} = \frac{|\mathcal{C}_1 \cap \mathcal{C}_2|}{|\mathcal{C}_1 \cup \mathcal{C}_2|}.$$

7.1.2 Model variants, hyperparameters and training duration

In our experiments we consider three models using the CNN-emb: ANNc combined with CNN-emb (CNN+ANNc), ANNr combined with CNN-emb (CNN+ANNr), and 3CosMul combined with CNN-emb (CNN+3CosMul). We do not present 3CosAdd, as it tends to have lower performance according to [LG14]. We also consider AE-emb as an AE model for the reconstruction task, as well as an embedding model for ANNr combined with AE-emb (AE+ANNr) and parallelogram rule combined with AE-emb (AE+par.). Additionally, we consider models that do not use any embedding model, described in Section 7.2, namely Lepage's Nlg toolkit (Nlg), Alea, and Kolmo.

CNN+ANNc is our main analogy detection model, together with the baselines Nlg, Alea@1, Alea@10, Kolmo@1, and Kolmo@10, described in **??**.

For analogy solving we use CNN+ANNr, CNN+3CosMul, AE+ANNr, and AE+par., as well as baselines Alea and Kolmo. We also use CNN+ANNc as a makeshift retrieval model, as mentioned in Remark 6.7 (Subsection 6.1.1). To do so, given an analogical equation $A : B :: C : x$ we use the same list of words used as candidates for retrieval models, and we solve the retrieval formula:

$$\text{ANNc retrieval}(A, B, C, \text{candidates}) = \underset{D \in \text{candidates}}{\text{argmax}} \ \text{ANNc}(e(A), e(B), e(C), e(D)) \tag{7.1}$$

where $\text{ANNc}(e(A), e(B), e(C), e(D)) \in (0, 1)$ is the score given by CNN+ANNc to the AP $A : B :: C : D$. In other words, we find the candidates that makes the analogical equation as "analogical" as possible according to CNN+ANNc.

Hyperparameter values. For CNN-emb, the only hyperparameter not fixed in Hyperparameter box 6.1 is the character embedding size m that we set to 64. Some of our early experiments, reported in Subsections 8.1.1 and 8.1.2 used $m = 512$ for Japanese, due to the larger number of characters. However, this exception was dropped as it did not have enough empirical grounding in terms of performance.

For AE-emb, we use one-hot vectors instead of character embeddings, so m is the number of characters of the training language (using the notation from Hyperparameter box 6.2). The target word embedding size is $|e(w)| = 256$.

For ANNc, the expected word embedding size is $n = 80$ in all our experiments due to the structure of CNN-emb. The first CNN layer uses $|f_1| = 128$ filters and the second $|f_2| = 64$ filters (using the notation from Hyperparameter box 6.2).

The expected word embedding size for ANNr is $n = 80$ when working with CNN-emb, and $n = 256$ when working with AE-emb. The parameters of layers f_1 and f_2 are not shared, as mentioned in Remark 6.9.

Training duration and optimization algorithm. For all models except AE-emb, we use Adam [KB15] for optimization with a learning rate of 10^{-3}, except when fine-tuning CNN-emb with CNN+ANNr, in which case a learning rate of 10^{-5} for CNN-emb gave better results.

To pre-train AE-emb, NAdam [Doz16] was used with a learning rate of 10^{-2} as it gave marginally better results than Adam. This pre-training for AE-emb lasts for 100 epochs or until the AE-emb $\mathcal{L}_{AE}$ on the development set stops improving.

We use a maximal training time of 20 epochs for CNN+ANNc, as it gave sufficiently good results without over-fitting. In addition to the pre-training of either CNN-emb or AE-emb, CNN+ANNr and AE+ANNr are trained for 50 additional epochs. For CNN+ANNr, unless specified otherwise, CNN-emb is frozen until $\mathcal{L}_{norm.random}$ stops improving, then unfrozen for the remaining epochs. A similar mechanism is integrated in $\mathcal{L}_{AE+ANNr}$.

7.1.3 Data in the training, development, and test sets

For our experiments using Siganalogies, we use the three sets of examples used in traditional DL(as explained in Subsection 3.1.2):

- the training set contains the examples used to train the model, *i.e.*, the training examples;
- the development set contains examples not seen during training and used to make decisions regarding model training: for instance, to identify overfitting and to interrupt the training when the performance stops increasing (the so called early stopping);
- the test set to measure the final performance, with examples that appear in neither the training set or development set.

Dataset split for analogy detection and analogy solving. For APs, we follow the training and test split from Sig16, Sig19, and JBATS when possible. The training set is split into training and development analogies. We exclude duplicates of the form $A : B :: C : D \leftrightarrow C : D :: A : B$, ad they will be generated by data augmentation, but we keep analogies of the form $A : B :: A : B$. Unless specified otherwise, we randomly sample 500 development and 5000 test APs, and at most 50000 training APs (before augmentation). Given a language from either Sig16, Sig19, or JBATS, the split between training, development, and test APs is the same across all experiments. However, the APs sampled within these sets may differ.

For some languages, such as Japanese, the number of APs is insufficient to sample 50000 training, 500 development, and 5000 test APs. In that case, we reduce the number of training examples. For comparability, we keep the number of training steps $n_{\text{steps}} = n_{\text{epochs}} \times n_{\text{training examples}}$ constant across all languages. As such, reducing the number of training examples will increase the number of epochs, resulting in a comparable training duration for all languages.

Languages with small test sets. Maltese, Navajo, and Turkish from Sig16 were tested on only 3707, 4843, and 11360 APs (before augmentation) respectively, due to the smaller size of the test dataset.

Dataset split for reconstruction task. For the pre-training of the AE-emb, we use words instead of APs. We consider all the words appearing in the original data from Sig16, Sig19 and JBATS as a dataset of words for the language, without distinction between the training set, development set, and test set. As such, this data includes words that do not appear in any of the APs selected for the training set, development set, or test set. We randomly sample 500 development and 500 test examples, and at most 40000 training examples (before augmentation), with no overlaps between the sets. If too few words are available, we use the procedure described at the end of the previous paragraph and reduce the training set.

7.2 Additional details for the symbolic baselines

We differentiate our three symbolic baselines Alea [LYZ09], Nlg [FL18], and Kolmo [Mur+20] from the vector-based baselines parallelogram rule, 3CosAdd, 3CosMul, as the latter use our embedding models. The symbolic baselines are introduced in Subsection 4.3.2, while the vector-based approaches are presented in Subsections 3.4.1 and 6.2.1.

In this section, we provide technical details on the symbolic baselines we use in this chapter. The Alea and Kolmo models are not designed for analogy detection, so we perform several adaptations described in Subsections 7.2.2 and 7.2.3 to use them for analogy detection.

Finally, in Subsection 7.2.4, we describe an issue we encountered regarding the time required to apply Alea and Kolmo on Siganalogies.

7.2.1 Analogy detection with Lepage's Nlg toolkit

We use the analogy classifier (`is_analogy` in `nlg/Analogy/tests/nlg_benchmark.py`) from Fam and Lepage's toolkit [FL18] to classify analogies in the same manner as with our DL model. The approach uses the algorithm defined in [Lep98], that we refer to as as Nlg. It is based on two processes described below. For further implementation details, refer to `verifnlg()` in `nlg/Analogy/C/nlg.c`.

Edit distance and longest common subsequence. To do analogy detection, Nlg first produces alignment matrices between two words w_1, w_2. The alignment matrix itself is not used for analogy detection, however the matrices are used to compute the edit distance *pdist* between w_1, w_2, using insertion (cost 0), replacement (cost 1), and deletion (cost 1). As insertion and deletion do not share the same cost, $pdist(w_1, w_2)$ may differ from $pdist(w_1, w_2)$. For instance, $pdist(\text{“}unlike\text{”}, \text{“}like\text{”}) = 2$ (2 deletions) while $pdist(\text{“}like\text{”}, \text{“}unlike\text{”}) = 0$ (2 insertions). The similitude is the length of their longest common subsequence, obtained with $sim(w_1, w_2) = |w_1| - pdist(w_1, w_2)$.

Constraint to fulfill to have a valid AP. To identify valid APs process is a constraint on the presence of characters, following the Distribution postulate (see Subsection 2.3.1) and formulated as $|A| \geq pdist(A, B) + pdist(A, C)$. If $|A| > pdist(A, B) + pdist(A, C)$, then some subsequences of A appear in both B and C, and therefore appear also in D by Exchange of the Extremes and Distribution. These subsequences are noted $common(A, B, C, D)$, and checking whether $A : B :: C : D$ holds is assimilated to checking if $|A| = pdist(A, B) + pdist(A, C) + common(A, B, C, D)$.

Remark 7.2

The approach in [FL18; Lep98] also defines an analogy solving algorithm, however we do not use it in our analogy solving experiments as it is outperformed by Kolmo in most cases, as shown by Murena, Al-Ghossein, et al. [Mur+20].

7.2.2 Analogy solving and analogy detection with Alea

As described in Subsection 4.3.2, Alea [LYZ09] uses random slicing and merging of the character strings A, B and C to obtain potential solutions to $A : B :: C : x$.

Principle of the approach. Two operations are used: a complement operation $w_1 \setminus w_2$ and a shuffle operation $w_1 \circ w_2$. Considering w_1, w_2 as sequences of symbols, $w_1 \circ w_2$ is defined by Langlais, Yvon, and Zweigenbaum as “the strings obtained by selecting (without replacement) alternatively in w_1 and w_2”, while keeping the relative order of the characters from w_1 and from w_2. For instance, “*DOG*” $\circ$ “*cat*” contains “*DOGcat*”,“*catDOG*”,“*cDatOG*”,“*DcOaGt*”, *etc.* For $w_1 \setminus w_2$, the characters from w_2 are removed from w_1, in a left to right manner, and $w_1 \setminus w_2$ is the set of all possible results. For instance, “*cactus*” $\setminus$ “*cat*” = {“*cus*”}.

Estimation of the likelihood of a solution. The Alea approach performs a Monte Carlo estimation of the most likely elements in $(B \circ C) \setminus A$, which are considered the most likely solutions to $A : B :: C : x$. The Monte Carlo estimation is performed by randomly sampling x' from $B \circ C$, then randomly sampling x from $x' \setminus A$. The random process is repeated s times, in our case $s = 1000$ following [LYZ09]. How frequently a particular word appears in the sample is an estimates of the likelihood in $(B \circ C) \setminus A$.

Variants of the model. For analogy solving, we use the most likely element obtained from the Monte Carlo estimation as the solution to the analogical equation. For analogy detection, we use two different variants to classify $A : B :: C : D$:

- **Alea@1:** if D the most likely solution to $A : B :: C : x$, the AP is classified valid, and invalid otherwise;
- **Alea@10:** if D appear in the 10 most likely solutions to $A : B :: C : x$, the AP is classified valid, and invalid otherwise.

Note that Alea@1 is biased in favor of invalid and Alea@10 in favor of valid APs, as if D is among the solutions in rank 2 to 10 the AP will be classified invalid with Alea@1 and valid with Alea@10.

7.2.3 Analogy solving and analogy detection with Kolmo

As mentioned in Subsection 4.3.2, Kolmo [Mur+20] estimates the least complex transformation f that is applicable on C and such that $B = f(A)$ to obtain potential solutions to $A : B :: C : x$.

Principle of the approach. The approach considers transformations f that are defined on subsets of all possible strings writable with an alphabet. If f is defined for A, C, then $A : f(A) :: C : f(C)$ is considered valid.

To estimate the Kolmogorov complexity of $A : f(A)$, a procedure producing both A and $f(A)$ is described by a sequence of instructions associated with a cost. The sum of all costs of these instructions is the estimated complexity. The set of instructions is defined as $\Sigma = \mathcal{A} \cup \mathbb{N} \cup \{gr, let, mem, ?\} \cup \{:, ::\}$, with $\mathcal{A}$ the alphabet of symbols manipulated by the language. Using this language, a procedure is constructed as a sequence of elements from Σ, written separated by commas. The instructions in the language can interact with a memory, structured as a heap.

The instruction let delimits a memory storage, which can contain variables noted with the instruction $?i$ for $i \in \mathbb{N}$, where i is the index of the variable. This memory storage is added to the memory heap, and can be accessed using the instruction mem, j for $j \in \mathbb{N}$, with j the depth of the instruction to retrieve in the heap. Right after the call to mem, j, the content of the variables is specified. For instance: $let, ?0, ?0, let$ defines a duplication operation, and calling $mem, 0, `a'$ will specify the variable $?0 = `a' \in \mathcal{A}$, resulting in aa. As many values are expected after mem, j as the number of variables in the instruction, if any. The instruction gr delimits a group of letters from $\mathcal{A}$, that is used as a single variable value for mem, j. For instance, after defining the duplication operation above, $mem, 0, gr, `a', `b', gr$ will result in $abab$, as $gr, `a', `b', gr$ corresponds to ab.

Finally, the operations are compounded to produce the form of $A : B :: C : D$, using the template $let, \ldots, :, \ldots, let$ to define the input and result of transformation f, followed by $mem, 0, \ldots, ::, mem, 0, \ldots,$ to define the input for $let, \ldots, :, \ldots, let$ that will produce $A : f(A)$ on one side, and $C : f(C)$ on the other. See examples of such series of instructions in [Mur+20, Figure 1]. To evaluate the complexity of f, the following costs are used: gr costs 2, any character from $\mathcal{A} \cup \{let, :, ::\}$ costs 3, mem, i and $?i$ both cost $i + 4$.

The approach was shown to outperform Alea and the analogy solving approach from [FL18; Lep98] (as mentioned in Remark 7.2) on the dataset Kakenhi-Sig16 [Lep17], see [Mur+20, Table 2].

Variants of the model. For analogy solving, we use the solution produced by the least complex transformation as the solution to an analogical equation. For analogy detection, we use two different variants to classify $A : B :: C : D$, following what we use for Alea:

- **Kolmo@1:** if D the solution to $A : B :: C : x$ produced by the least complex transformation, the AP is classified valid, and invalid otherwise;
- **Kolmo@10:** if D appears in the 10 most likely solutions to $A : B :: C : x$, the AP is classified valid, and invalid otherwise.

To obtain the 10 most likely solutions to $A : B :: C : x$, the likelihood of a solution w is estimated as $l(w) = \sum_{f \in F_w} 2^{|f|}$, for F_w the set of transformations that produce w as a potential solution to $A : B :: C : x$, and $|f|$ the total cost of the instructions in f. Similarly to Alea@1 and Alea@10, Kolmo@1 is biased in favor of invalid and Kolmo@10 in favor of valid APs.

7.2.4 Run time issues on Alea and Kolmo

On some examples from Siganalogies, Alea and Kolmo require a significantly longer time compute the solution to the analogical equation, with most examples requiring less than a second to be solved while a few others require significantly more than 10 seconds.

In particular, for Sig16, we identified that the length of words (in particular, Finnish and German) and the number of repeated adjacent letters (in particular, Finnish and Navajo) correlate with this slowdown. As both characteristics are particularly present in Finnish, this language presented the most significant slowdown of Alea and Kolmo, and in particular Alea@10 and Kolmo@10, that require a larger amounts of estimates to obtain the 10 most likely words.

To circumvent this issue, in our first experiments [Als+21a; Als+21c; MMC22], for the most problematic language of Sig16 (Finnish), we reduced the number of examples used for the evaluation of Alea and Kolmo and excluded Alea@10 and Kolmo@10. In later experiments, to avoid stalling the obtention of results, we interrupt the analogy solving and analogy detection processes that take longer than 10 seconds and consider that the baseline failed to solve the analogical equation. In terms of analogy detection, this corresponds to an invalid AP. The timeout, introduced in [Mar+22a], allows to compare more fairly Alea and Kolmo with the other approaches, as we use the same test examples.

Details of the portion of examples resulting in timeout observed on the test data are reported in Subsection 7.2.4. From the low timeout rate, it can be seen that the timeout process minimally impacts the measured performance.

	Alea	Kolmo
Arabic	–	0.039%
Finnish	0.040%	3.802%
Georgian	–	0.026%
German	0.023%	1.998%
Hungarian	0.002%	0.102%
Maltese	–	0.330%
Navajo	–	0.018%
Russian	–	0.334%
Spanish	–	0.188%
Turkish	0.028%	0.496%
Japanese	–	–

Table 7.1: Extract from Table 2 from [Mar+22a]. Timeout rate of Alea and Kolmo on the test sets of Sig16 and JBATS. When nothing is specified ("–"), the timote rate is 0.

7.3 Analogy detection performance

This section details our experiments on analogy detection. The performance of the models using the most up to date training procedure and of the baselines mentioned in Section 7.2 is detailed in Subsection 7.3.1, while the experiments that led to this training procedure are reported in Subsections 7.3.2 and 7.3.3. We also explore the performance of the CNN+ANNc model when transferred to different data from the same language in Subsection 7.3.4, leveraging redundant languages between Sig16 and Sig19.

7.3.1 General observations

In [Als+21a; Als+21b], we performed experiments with the data augmentation process with 8 valid and 3 invalid permutations for each training example, as described in Subsection 6.3.1. We later reproduced these experiments with using the sampling described in Subsection 6.3.3 to obtain better overall performance. The results are reported in Table 7.2. Only the results for 8 valid and 8 invalid permutations are presented in this subsection, see Subsection 7.3.3 for the results with the 8 valid and respectively 3 and 24 invalid permutations.

Results and discussion for CNN+ANNc. In the results presented in Table 7.2, the CNN+ANNc model significantly outperforms the best baselines for valid analogies (paired Student t-test $p \approx 0.00027$) and there is no significant difference for invalid analogies (paired Student t-test $p \approx 0.869$). For each language, we observe very significant differences between CNN+ANNc and the baseline a p-value $p < 10^{-10}$ for the 1-sampled t-test.

Using CNN-emb, ANNc manages to capture the features of morphological APs necessary for analogy detection. Results on transferability experiments Section 7.6 and Chapter 8 also indicate that the processing of AP itself is correctly embedded in the classifier part, while the morphological information dependant on the language appears to be mostly encoded in the embedding model.

Baseline performance. In Table 7.2, the reported results are taken from [Als+21a]. As mentioned in Subsection 7.2.4, the experiments in [Als+21a] were performed without timeout. Instead, for Finnish, Alea@1 and Kolmo@1 were run on 8% of the data (4000 instead of 50000 base APs), while Alea@10 and Kolmo@10 where not applied (see Subsection 7.2.4 for a discussion). We noticed in [Als+21a] that between the 5 baselines (Nlg, Alea@1, Alea@10, Kolmo@1, and Kolmo@10), the

	ANNc (ours)		**Best Baseline**	
	Valid APs	Invalid APs	Valid APs	Invalid APs
Arabic	**99.39**	97.40	34.21 (Alea@10)	**97.79** (Kolmo@1)
Finnish	**99.58**	97.98	25.60 (Nlg)	***98.78*** (Alea@1)
Georgian	**99.87**	92.06	93.20 (Kolmo@10)	**95.21** (Alea@1)
German	**99.42**	95.24	86.90 (Alea@10)	**97.19** (Alea@1)
Hungarian	**99.84**	**98.71**	36.80 (Kolmo@10)	98.40 (Kolmo@1)
Maltese	**99.88**	**77.79**	78.05 (Alea@10)	69.29 (Kolmo@1)
Navajo	**99.00**	93.50	21.45 (Kolmo@10)	**94.93** (Kolmo@1)
Russian	**96.89**	89.96	42.37 (Alea@10)	**93.88** (Nlg)
Spanish	**99.69**	82.03	85.90 (Alea@10)	**86.62** (Nlg)
Turkish	**99.70**	**91.93**	44.76 (Alea@10)	91.40 (Kolmo@1)
Japanese	**100.00**	**98.61**	19.20 (Kolmo@10)	98.13 (Nlg)

Table 7.2: Table 2 from [Als+21c]. Classification accuracy (in %) on the test sets of Sig16 and JBATS, against the best performing baseline. For Finnish, Alea@1 and Kolmo@1 were run on 8% of the data (4000 instead of 50000 base APs, result indicated in italics), while Alea@10 and Kolmo@10 where not applied.

classifier Nlg is the best in only 4 of 22 cases[1]. This result was not expected, since the method has been developed for analogy detection, while Alea and Kolmo are analogy solving approaches that we adapted in a potentially faulty manner to analogy detection.

For all baselines, we notice a striking imbalance of performance between valid and invalid APs, with higher performance on negative permutations. This indicates a tendency of the models to be biased against valid APs, and is consistent with how Alea and Kolmo are designed, and how we adapted them for analogy detection. If the fourth element of an AP is not sufficiently straightforward, if the AP appears to breach the postulates of APs, or if the transformation involved is not regular enough, the symbolic baselines will consider it invalid. One such AP in English would be ("*build*", past tense):("*build*", past participle)::("*go*", past tense):("*go*", past participle). The corresponding words are "*built*":"*built*"::"*went*":"*gone*", which appears to breach Strong Identity and involves the irregular verb "*go*".

Analysis of where each model fails lead us to testing the baselines on the APs before applying the augmentation process. For Kolmo, the results were significantly different (Student t-test p-value < 0.05) before and after data augmentation, but not for Alea and Nlg. That Kolmo performs better on base forms hints that this approach is not resilient to the permutations performed when augmenting data. In particular, it tends to fail when Exchange of the Means is involved. This is not unexpected, as Kolmo is not designed from the postulate of APs.

7.3.2 Preliminary experiments on the emb model

Our first experiments using the ANN framework followed closely [LPR21], using pre-trained word embeddings models for analogy detection on Siganalogies. In particular, we used GloVe [PSM14], Word2Vec [Mik+13], and FastText [Boj+17] at the time (see Subsection 3.3.4 for a description of the three approaches).

We recently reproduced these experiments on the English data of Sig19, with the current, more robust version of ANNc. For the pre-trained semantic embeddings, we used GloVe6B[2] and FastTextWiki[3]. For GloVe6B, we use the models with embeddings of 50, 100, 200, and 300 dimensions, and the FastTextWiki embeddings have 300 dimensions. We did not include Word2Vec for two main reasons: its performance is comparable to the one of FastText and GloVe, for instance [DWW22]; like GloVe, Word2Vec is purely a distributional word embedding that does not encode sub-word information. As opposed to GloVe and Word2Vec, FastText contains information about sub-words, *i.e.*, strings of characters frequently appearing in words, which is remotely similar to the notion of morpheme. We did not consider more recent large scale models, such as BERT [Dev+19], as they

[1] 1 of 11 cases on valid APs and 3 of 11 cases on invalid APs.
[2] GloVe trained on a Wikipedia dump, see `https://nlp.stanford.edu/projects/glove/`
[3] `https://dl.fbaipublicfiles.com/fasttext/vectors-wiki/wiki.en.vec`

compute the embedding of words in their context while Siganalogies contains words out of context. The results are reported in Tables 7.3 and 7.4.

A first limitation we encountered is that the pre-trained word embedding models are not able to handle the full extent of the Siganalogies dataset. The words they can handle are limited by the words seen during training, even if the sub-word information in FastText brings more flexibility: 37.69% of the vocabulary of words in the English data is covered by GloVe6B against 53.35% for FastTextWiki. Taking only the words covered by both models reduces the vocabulary to 37.15% of its original size, and analogies containing only covered words represent 10.33% of the original data.

There is also a second, more fundamental limitation with semantic word embeddings. Assuming they only encode semantic information, semantic word embeddings do not contain the information necessary to tackle morphological tasks. For instance, they cannot differentiate the morphology of synonyms, for which the semantic information is the same but not the word forms (and thus the morphemes). However, in English (as in many languages) some semantic features translate directly to morphemes, for instance, the plural which corresponds to the morpheme "*-s*". As such, we can still expect some amount of success from GloVe6B and FastTextWiki, that we observe in Tables 7.3 and 7.4.

For a given task, embedding models in which the information to solve the task is easier to access will result in higher performance, in particular when a simple model like ANNc is used. Semantic word embedding models are geared towards semantics, so it would not be surprising that the morphological information they contain is not directly accessible. To handle this third limitation, we add a fully-connected layer after the pre-trained embedding model, meant to handle fine-tuning for the embedding model, and set its output dimension to 80. In that manner, we also obtain results that are more comparable to the CNN+ANNc model (see Subsection 7.1.2): the embeddings are of the same dimension, and both model can fine-tune the embedding for the analogy detection task. In Tables 7.3 and 7.4 the models with a "+" sign are equipped with a fully-connected layer.

As one could expect, models using pre-trained embeddings and trained and tested on non-covered analogies have a significantly lower performance than CNN+ANNc, as shown in Table 7.3, for the most part due to unrecognized words during training and testing. In this setting, there is no significant difference between the models, with improvements of 2% to 4% on the F1 when learning a fully-connected layer after the embedding. If we consider only covered APs for training and testing the model, the F1 for GloVe6B jumps to 94% to 99% for models with the fully-connected layer and to 81% to 84% for models without. The corresponding results are reported in Table 7.4. Surprisingly, FastTextWiki, which is based on embeddings of sub-words (which are similar to morphemes), achieves a lower F1 of 52% without and 56% with the fully-connected layer.

In addition to the advantages of character-based embeddings (avoiding unknown words) and to the performance gap we can see in Tables 7.3 and 7.4, the morphological embedding approaches described in Section 6.1 result in more lightweight embedding models, since it is not necessary to store an embedding for each word (or each sub-word for FastText). For instance, our CNN-emb model for English weights less than 100 Kilobyte (KB), while GloVe6B weights 171 Megabyte (MB), 347 MB, 693 MB and 1 Gigabyte (GB) for 50, 100, 200, and 300 dimensions, respectively, and FastTextWiki weights 6.6 GB.

7.3.3 Balancing the data for analogy detection, with CNN+ANNc

Early analogy detection experiments in [Als+21a] identified an imbalance in performance between valid and invalid permutations after data augmentation. To address this phenomenon, we experimented with data augmentation for analogy detection in [Als+21c]. The experiments were performed on the 10 languages from Sig16 as well as Japanese from JBATS using CNN+ANNc, with its default hyperparameters as stated in Subsection 7.1.2. Three training settings were considered to cover both over- and under-representation of valid against invalid permutations, as well as a more balanced setting:

- in setting 8/3 we use all 8 valid permutations, but only the 3 invalid permutations computed from the base AP;
- in setting 8/24 we use all 8 valid permutations and all 24 corresponding invalid permutations, including those computed from the valid permutations of the base AP;

Model	F1	Balanced accuracy	Accuracy on Valid APs	Accuracy on Invalid APs
FastTextWiki	52.21 ± 0.34	66.99 ± 0.28	59.69 ± 1.34	74.30 ± 1.44
GloVe6B 50	52.61 ± 0.21	67.30 ± 0.19	60.43 ± 2.33	74.17 ± 2.55
GloVe6B 100	52.82 ± 0.21	67.48 ± 0.18	60.05 ± 1.66	74.90 ± 1.78
GloVe6B 200	52.88 ± 0.33	67.52 ± 0.25	60.57 ± 1.96	74.47 ± 1.92
GloVe6B 300	52.70 ± 0.24	67.38 ± 0.21	60.07 ± 1.57	74.68 ± 1.86
FastTextWiki+	56.22 ± 0.35	70.08 ± 0.26	61.58 ± 1.75	78.58 ± 1.70
GloVe6B 50+	54.38 ± 0.34	68.70 ± 0.26	59.82 ± 1.37	77.57 ± 1.47
GloVe6B 100+	55.20 ± 0.25	69.31 ± 0.19	60.89 ± 1.10	77.74 ± 0.89
GloVe6B 200+	55.54 ± 0.34	69.59 ± 0.24	62.46 ± 1.53	76.72 ± 1.78
GloVe6B 300+	55.96 ± 0.24	69.92 ± 0.18	63.03 ± 1.71	76.81 ± 1.66
CNN+ANNc	**99.39** ± 0.04	**99.70** ± 0.05	**99.76** ± 0.12	**99.64** ± 0.03

Table 7.3: Appendix Table A1 from [MC24]. Original caption: "Performance of the ANNc model on analogy detection (in %, mean ± std.) on English (Sig19), for all analogies. We consider several pre-trained embedding variants, for 5 random initialization of the model. Models with a "+" sign have a fully-connected layer after the embedding and before ANNc. [. . .]"

Model	F1	Balanced accuracy	Accuracy on Valid APs	Accuracy on Invalid APs
FastTextWiki	52.14 ± 0.19	66.93 ± 0.11	60.06 ± 2.14	73.79 ± 2.03
GloVe6B 50	81.70 ± 1.43	89.81 ± 0.80	95.30 ± 0.90	84.32 ± 1.88
GloVe6B 100	84.56 ± 1.27	91.26 ± 0.69	94.51 ± 0.55	88.02 ± 1.42
GloVe6B 200	84.03 ± 0.46	90.95 ± 0.28	94.36 ± 0.55	87.54 ± 0.62
GloVe6B 300	82.50 ± 0.42	90.09 ± 0.47	94.35 ± 2.52	85.83 ± 1.73
FastTextWiki+	56.24 ± 0.30	70.08 ± 0.24	61.08 ± 1.62	79.09 ± 1.44
GloVe6B 50+	94.83 ± 0.86	97.15 ± 0.47	97.77 ± 0.52	96.52 ± 0.68
GloVe6B 100+	98.37 ± 0.17	99.06 ± 0.11	99.11 ± 0.30	99.01 ± 0.19
GloVe6B 200+	98.76 ± 0.15	99.34 ± 0.10	99.51 ± 0.18	99.17 ± 0.12
GloVe6B 300+	98.83 ± 0.15	99.36 ± 0.09	99.47 ± 0.19	99.25 ± 0.14

Table 7.4: Appendix Table A2 from [MC24]. Original caption: "Performance of the ANNc model on APs detection (in %, mean ± std.) on English (Sig19), for covered APs only. We consider various variants of pre-trained embeddings, for 5 random initialization of the model, **with all models trained and tested with covered APs only**. Models with a "+" sign have a fully-connected layer after the embedding and before ANNc. [. . .] "

- in setting 8/8: we use the sampling described in Subsection 6.3.3 to obtain 8 valid permutations and 8 randomly sampled invalid permutations.

For testing, we use the default setting described in Subsection 7.1.2, which corrsponds to setting 8/24. We also distinguish the performance on valid (+) and invalid (−) APs. In [LPR19; LPR21], the 8/24 was used. For training, we used the 8/24 setting in [Als+21a; Als+21b; Als+21c] and the 8/8 setting in [MC24; MMC22]. For testing, the 8/24 setting was used in all of our work.

	8/3		8/24		8/8	
Language	+	−	+	−	+	−
Arabic	**99.89**	97.52	95.66	**99.39**	99.39	97.40
Finnish	99.44	82.62	83.46	96.94	**99.58**	**97.98**
Georgian	99.83	91.71	88.89	**99.53**	**99.87**	92.06
German	**99.48**	89.01	67.20	87.26	99.42	**95.24**
Hungarian	**99.99**	98.81	98.40	**99.25**	99.84	98.71
Maltese	99.53	90.82	89.29	**97.40**	**99.88**	77.79
Navajo	97.95	79.85	46.33	**97.40**	**99.00**	93.50
Russian	**99.94**	78.33	72.66	**99.80**	96.89	89.96
Spanish	99.48	92.63	82.95	**99.35**	**99.69**	82.03
Turkish	**99.99**	**98.65**	100.00	98.64	99.70	91.93
Japanese	99.96	77.83	37.91	**98.85**	**100.00**	98.61

Table 7.5: Accuracy results (in %) for the classification task for valid (+) and invalid (−) APs, with 3 distinct training settings (8/3, 8/24, 8/8). Bold values are the best performance on valid (+) and invalid (−) APs respectively.

		8/8	8/24
+	8/3	$p = 1.05 \times 10^{-1}$ ($t = 1.78$)	$p = 6.40 \times 10^{-3}$ ($t = 3.43$)
	8/8	—	$p = 7.11 \times 10^{-3}$ ($t = 3.37$)
−	8/3	$p = 5.93 \times 10^{-13}$ ($t = -45.80$)	$p = 8.29 \times 10^{-10}$ ($t = -22.04$)
	8/8	—	$p = 5.72 \times 10^{-2}$ ($t = -2.15$)

Table 7.6: Paired Student t-test on the evaluation for valid (+) and invalid (−) APs, with 3 distinct training settings (8/3, 8/24, 8/8). t is the test variable.

From settings 8/3 and 8/24, where invalid APs are respectively under- and overrepresented during training, we observe that overrepresented leads to a drop in the performance of the opposing class. In particular, an imbalance in favor of valid APs leads to very good results on valid APs and poorer results on invalid APs, and conversely, an imbalance in favor of invalid APs leads to very good results on invalid APs and poorer results on valid APs.

To maximize performance on both valid and invalid APs, we decided to balance the amount of training data for both classes. Random sampling of 8 invalid APs from the 24 available allows balancing while keeping a comparable representation of invalid permutations, As mentioned in Subsection 6.3.3, 8 invalid APs are sampled from the 24 available. On average on the whole training set, this process maintains a similar representation of each of the 24 invalid permutations. The results of this balanced setting 8/8 correspond to what was expected, as CNN+ANNc obtains results comparable with the best of settings 8/3 and 8/24 for valid and invalid APs respectively. The p-values mentioned hereafter come from paired Student t-tests, reported in Table 7.6. On valid APs, setting 8/8 produced significantly better results than setting 8/24 ($p < 0.01$) where valid APs are underrepresented, with no significant difference with setting 8/3 ($p > 0.1$). On invalid APs, setting 8/8 was significantly better than setting 8/3 ($p \ll 0.01$). There is a weakly significant performance difference between settings 8/8 and 8/24 ($0.05 < p < 0.1$), with 8/24 performing slightly better overall except on Finnish and German.

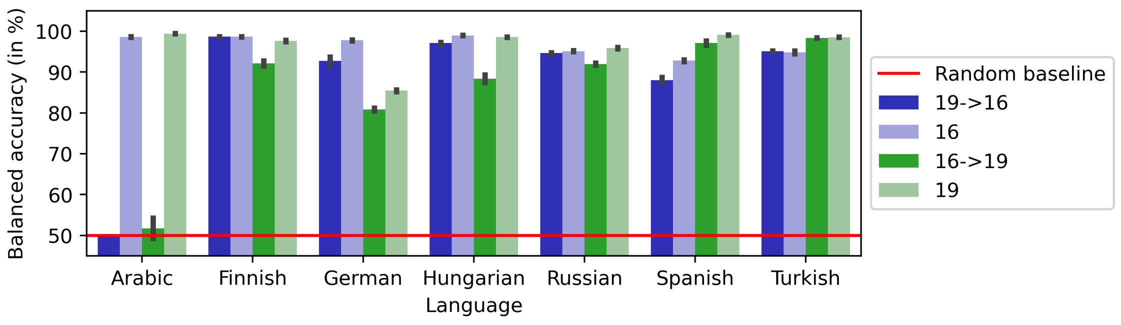


Figure 7.1: Figure 2 from [MMC22]. Accuracy of the transferred model compared to the model trained on the target setting.

7.3.4 Ablation study on the transfer performance between Sig16 and Sig19

In Siganalogies, 7 languages appear in both Sig16 and the high-ressource languages of Sig19, namely: Arabic, Finnish, German, Hungarian, Russian, Spanish, and Turkish. As Sig16 and Sig19 do not contain the same data, this configuration allowed us to confirm the extent of the generalization ability of CNN+ANNc in [MMC22]. In particular, we confirmed that in most cases, the model successfully transfers between closely related domains with a slightly different distribution of morphological transformation.

Experimental setting. We report in Figure 7.1 the balanced accuracy (bal. acc.) in 4 settings. On the one hand, settings **16** and **19** correspond to models trained and tested on the same dataset, respectively on the data from Sig16 and Sig19. This is the best performance available as the model was trained on the same distribution as the test set, and forms out topline. On the other hand, **16** $\rightarrow$ **19** and **19** $\rightarrow$ **16** correspond to models trained and tested on the different datasets, using the “training setting $\rightarrow$ test setting” notation. For **16** $\rightarrow$ **19** the models are trained on Sig16 and tested on Sig19, and the opposite for **19** $\rightarrow$ **16**.

We report the performance of a random baseline, that would randomly answer valid or invalid AP with equal chances. As we use balanced accuracy, this is equivalent to a majority baseline answering valid for all inputs, or invalid for all inputs. When performing transfer, we transfer both the CNN-emb and ANNc without fine-tuning. For each setting, 10 random initializations are trained in the default setting for CNN+ANNc described in Subsection 7.1.2.

Results and discussion. The performance for the transferred model is comparable to or slightly lower than the performance obtained by the topline, for both transfer directions. The only exception is Arabic, where the performance drops to that of the random baseline after transfer.

In Subsection 5.5.1, we mentioned the limitation of character representation which is inconsistent between Sig16 and Sig19. As we transfer both the CNN-emb and ANNc without fine-tuning, we do not adapt the character embeddings in CNN-emb and any unknown character is represented with a character embedding full of 0s (the same value is used when padding). As one could expect, this negatively impacts the performance of the model.

To be more accurate, a significant correlation can be noticed between the performance of the transferred model and the character coverage of the target domains by the source domains, *i.e.*, the number of characters present in both domains divided by the number of characters present in the target domain, reported in Figure 7.2. In particular, for the raw accuracy of the transferred model, we observe a Pearson correlation coefficient with coverage of $r = 0.9639$ for 19 $\rightarrow$ 16 and $r = 0.7595$ for 16 $\rightarrow$ 19. When normalizing the transfer performance by the performance trained on the target domain, the correlation goes up to $r = 0.9739$ for $\frac{19\rightarrow16}{16}$ and $r = 0.8639$ for $\frac{16\rightarrow19}{19}$. A critical case of this correlation can be seen for Arabic, which is romanized in Sig16 but uses UTF8 Arabic characters in Sig19, leading to a coverage close to 0%. These results identify the embedding model as the main factor limiting in the transfer performance in this experiment.

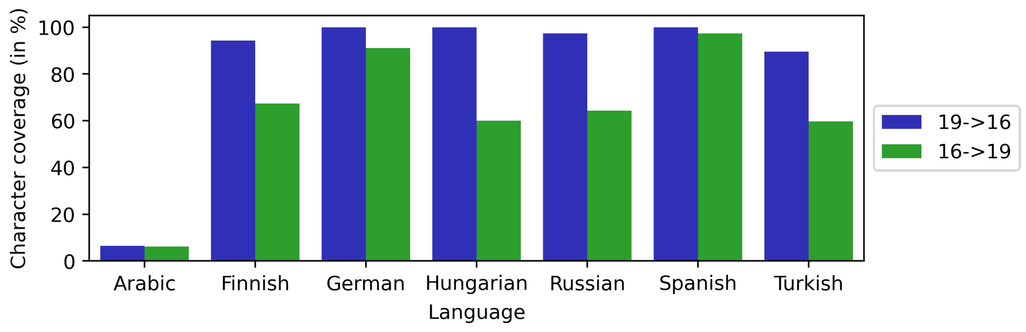


Figure 7.2: Figure 2 from [MMC22]. Coverage of the target language characters by the source language characters.

7.4 Analogy solving performance

This section details our experiments on analogy solving. The performance of the models using the most up to date training procedure and of the baselines mentioned in Section 7.2 is detailed in Subsection 7.4.1, while the experiments that led to this training procedure are reported in Subsections 7.4.2 and 7.4.4. The results discussed in this section are, for the most part, on a subset of 16 languages: 3 from Sig16 and 13 from Sig19. These languages were chosen based on the performance of the AE-emb model, described in Subsection 7.4.3. We conclude the section with a case study of the analogy solving results on Navajo and Georgian from Sig16, reported in Subsection 7.4.5.

7.4.1 General observations

To evaluate the performance of all analogy solving models, we use the accuracy (see Subsection 7.1.1). For retrieval models, this corresponds to retrieval hit rate at 1, and hit rate at k for $k \in \{1, 3, 5, 10\}$ are detailed in Tables A.1 and A.2 of Appendix B. For generative models, we use word accuracy by taking only the most likely prediction for each character. The retrieval and generative models performance are reported in Table 7.7.

General comments. The embedding based approaches outperform the symbolic baselines in all settings using ANNr and in most settings with CNN+3CosMul, CNN+ANNc, and AE+par.. This result is not surprising, as there is a well known trade-off between the performance of deep learning and the explainability of symbolic approaches. It also matches what was obtained for analogy detection (see Subsection 7.3.1).

Comparison with the symbolic baselines. The baseline generation models, Alea and Kolmo, have very low performance on Arabic while this is not the case for DL models. As mentioned in Subsection 5.5.1, in Sig19 the UTF8 character encoding for Arabic results in longer and more complex sequences of characters than one could expect when reading the text. For Slovene, Alea and Kolmo have similarly very low performance, but the low performance cannot be attributed to character encoding. Indeed, characters used in Slovene are more or less the same as for Portuguese and English, two languages where Alea and Kolmo performed much better. Therefore, it is more likely that the inflectional morphological transformations in Sig19 for those languages are harder to handle for Alea and Kolmo than for DL models, as the latter can learn to adapt to the morphological quirks of each language.

Interestingly, English and Portuguese, where AE+par. performs the worst, are also the languages of Sig19 where the symbolic baselines Alea and Kolmo perform the best. Portuguese has a very regular inflectional morphology [Bra81], and the design of Alea and Kolmo allows them to handle morphological transformations which are frequent in those two languages, in particular affixation [Bra81; LYZ09; Mur+20]. Given that the performance of AE+ANNr and of models using the CNN-emb embedding does not show the same tendency as AE+par., we hypothesize that the AE-emb together with the reconstruction task for pre-training has trouble handling those

Language	Retrieval models			Generative models			
	CNN+ANNr	CNN+3CosMul	CNN+ANNc	AE+ANNr	AE+par.	Alea	Kolmo
			Sig16				
Georgian	**97.60** ± **0.23**	85.58 ± 7.37	76.77 ± 9.57	**87.50**± **2.08**	**87.06** ± **6.28**	84.97	79.94
Hungarian	**89.06** ± **1.71**	74.89 ± 5.27	72.95 ± 8.02	**90.58**± **0.63**	83.57 ± 6.63	35.24	32.07
Turkish	**84.75** ± **2.04**	52.42 ± 4.33	44.43 ± 13.95	**79.81**± **8.58**	**83.03** ± **14.07**	42.09	39.45
			Sig19				
Adyghe	**93.37** ± **0.97**	58.01 ± 8.66	80.11 ± 5.10	**98.50** ± **0.25**	81.90 ± 7.35	47.94	31.25
Arabic	**72.08** ± **3.49**	21.67 ± 5.44	40.73 ± 9.93	**83.99** ± **1.28**	78.40 ± 12.37	2.21	3.34
Bashkir	57.63 ± 5.48	37.76 ± 11.12	**65.38** ± **6.01**	**95.15** ± **0.92**	80.38 ± 18.55	22.29	29.89
English	**92.29** ± **0.91**	66.83 ± 15.67	65.51 ± 6.25	**92.29** ± **4.57**	65.61 ± 19.96	60.15	47.69
French	**93.15** ± **0.96**	80.37 ± 7.97	62.87 ± 17.65	**88.25** ± **2.24**	76.04 ± 10.20	54.48	54.39
Hebrew	**66.52** ± **2.59**	15.47 ± 10.12	36.10 ± 4.28	**92.50** ± **0.25**	**91.16** ± **7.29**	19.50	16.17
Portuguese	**93.12** ± **1.10**	58.52 ± 18.48	70.53 ± 9.88	**93.11** ± **8.32**	62.88 ± 24.26	78.01	71.28
Sanskrit	**64.18** ± **2.62**	33.20 ± 9.34	42.59 ± 4.88	**91.48** ± **0.78**	83.83 ± 5.37	42.80	28.83
Slovak	**56.23** ± **4.57**	49.43 ± 3.13	39.58 ± 3.37	**78.90** ± **0.86**	82.62 ± 6.66	30.66	28.81
Slovene	**71.99** ± **2.24**	57.79 ± 8.59	51.88 ± 6.69	**82.41** ± **10.98**	**80.95** ± **8.19**	2.64	5.43
Swahili	**68.56** ± **6.09**	44.84 ± 7.77	44.46 ± 3.85	**97.49** ± **0.21**	81.94 ± 21.80	60.23	43.02
Welsh	**63.80** ± **3.13**	47.30 ± 4.67	47.58 ± 5.72	**96.79** ± **0.23**	87.62 ± 12.69	14.47	19.15
Zulu	**76.59** ± **2.65**	58.53 ± 4.42	42.56 ± 6.55	**93.42** ± **0.73**	81.96 ± 15.81	26.17	27.69

Table 7.7: Appendix Table A5 from [MC24]. Accuracy (rate of fully well predicted words, in %) on the analogy solving tasks. For models depending on an embedding model, and thus sensitive to random initializations, we report performance as mean ± standard deviation over 10 random initialization in each setting. Symbolic baselines were tested in the same setting as our models.

transformations. Nevertheless, due to the high variance in the performance it is hard to draw definite conclusions.

Note that there is a significant difference in the performance of AE+par. depending on the permutation of the base AP. In particular, the performance of AE+par. is higher when the solution to the analogical equation is the same one of the other three elements, namely, analogical equations based on Identity ($A : A :: B : x, x = B$) or Reflexivity of Conformity ($A : B :: A : x, x = B$). This is due to the nature of the parallelogram rule, for which we always have $x = e(B)$ as the solution to $e(A) : e(A) :: e(B) : x$ and $e(A) : e(B) :: e(A) : x$, *i.e.*, Identity and Reflexivity of Conformity hold by definition. As no error can appear in the analogy solving part, the performance of AE+par. depends exclusively on the performance of the AE-emb for this kind of examples.

ANNr improves analogy solving performance. The CNN+3CosMul, CNN+ANNc, and AE+par. models have significantly lower performance on analogy solving than CNN+ANNr and AE+ANNr. While it is possible that this performance gap is caused by the fine-tuning on analogy solving present when ANNr is used but not the other models, we argue that this is not the main factor. Indeed, CNN+ANNc and CNN+3CosMul use an embedding model trained with analogy while AE+par. does not, yet the latter outperforms the former. From this observation, ANNr and the corresponding training procedure are likely to perform better on analogy solving than the other approaches we consider.

Analogical training reduces sensitivity to random initialization. We observe in Table 7.7 the using ANNr reduces the standard deviation on performance, in particular compared to AE+par.. Namely, the AE-emb has a large standard deviation on the reconstruction task (see Subsection 7.4.4), emphasized by the 50 initialization used to compute performance. A similar observation can be made for CNN+3CosMul, CNN+ANNc, and AE+par., while CNN+ANNr and AE+ANNr have a significantly lower standard deviation. What separates the two groups of models is the use of ANNr and the fine-tuning of the embedding models using analogical data augmentation.

AE+ANNr outperforms CNN+ANNr despite the Open World Assumption. As can be seen in Table 7.7, we obtain equivalent or better performance with AE+ANNr than we obtain with CNN+ANNr, even though the latter benefits from an embedding model designed specifically for morphology and from a closed set of possible solutions to retrieve from.

AE+ANNr differs from CNN+ANNr mostly in that AE+ANNr uses of the decoder output to compute the loss. Having direct access to the characters of the generated solution makes it likely that AE+ANNr learns to avoid small differences in the word form. Conversely, due to the barrier of the retrieval process, those small differences are not directly accessible with the CNN-emb embedding model.

Using ANNc for analogy solving. ANNc is designed for analogy detection and has a tendency to distinguish poorly between comparatively meaningful solutions to the analogical equation. This can be confirmed by extending the results to the top 10 highest classification scores, which significantly increases the accuracy, as discussed in a later paragraph.

Despite this limitation, CNN+ANNc outperforms CNN+ANNr on Bashkir, and CNN+3CosMul on 7 languages (Adyghe, Arabic, Bashkir, Hebrew, Portugese, Sanskrit, and Welsh). As mentioned before, the results become even more interesting when we look at the performance within the top 3, 5, and 10 retrieved words, for which CNN+ANNc outperforms CNN+3CosMul respectively on 8, 11, and 12 languages out of 16. This is in line with the properties of ANNc and 3CosMul, as the latter is a fixed formula while the former is trained together with its embedding model. Not only ANNc is more flexible as it is learned while 3CosMul is a fixed formula, CNN+ANNc has a better adequation between the embedding space and the analogy model as they are trained together.

Nevertheless, ANNc is significantly slower than the other retrieval approaches. For instance, for Turkish from Sig16, CNN+ANNc took more than 20 minutes to retrieve the solutions of the augmented test set against 46 seconds for CNN+3CosMul and 40 seconds for CNN+ANNr. Furthermore, on a computer equipped with an Nvidia RTX A5000 Mobile used at close to 100% of its processing capabilities, training took 2.5 minutes for CNN+ANNc, for CNN+3CosMul we reuse the CNN-emb from CNN+ANNc so the training time is the same, and we need an additional 2 minutes to train CNN+ANNr. From this example, the time required to retrieve the solution with

Language	Cosine similarity	Euclidean distance
Arabic	51.41	**51.53**
Finnish	**72.84**	72.23
Georgian	93.37	**93.44**
German	87.55	**87.78**
Hungarian	**68.31**	68.13
Maltese	75.68	**76.94**
Navajo	**45.81**	47.41
Russian	**69.57**	69.05
Spanish	**87.86**	87.53
Turkish	**70.10**	67.77
Japanese	**19.76**	17.50

Table 7.8: Accuracy results (in %) for preliminary experiments on analogy solving with CNN+ANNr with Euclidean and cosine distance for retrieval. From the internship report of Safa Alsaidi, Amandine Decker, and Puthineath Lay.

CNN+ANNc is close to 10 times longer than the time to train a CNN+ANNr model from the CNN+ANNc model and then apply it. Using CNN+3CosMul is even faster as it is not necessary to train a new model, reaching a 30 times speedup. The slowness is caused by having to repeat the computation of the analogy detection score for all the words of the vocabulary, and that for each analogy to solve. Careful engineering might reduce the impact on run time. For example, selecting a subset of candidates using CNN+3CosMul before retrieving the most suitable on with CNN+ANNc could allow for a significant speedup.

Looking a bit further from the "best" answer In Tables A.1 and A.2 of Appendix B, we extend the retrieval further than the closest solution (*i.e.*, the accuracy or hit rate at 1) to the hit rate at k for $k \in \{1, 3, 5, 10\}$, for CNN+ANNr, CNN+3CosMul, and CNN+ANNc. This experiments were initialy published in [MC24].

For all languages, increasing the retrieval threshold k increases the performance of all retrieval models. CNN+ANNr rapidely reaches above 99% hit rate in languages from Sig16. This corresponds to an improvement of 3% for Georgian, 10% for Hungarian, and 15% for Turkish. Similar improvements can be observed in languages from Sig19, with the minimum hit rate at $k = 10$ around 95%, in Swahili. For languages in Sig16, the increase in performance brings CNN+ANNc closer to the performance of CNN+3CosMul, and for languages from Sig19, CNN+ANNc systematically outperforms CNN+3CosMul at $k = 10$. For the only language in which CNN+ANNr did not get the best results, namely Bashkir, increasing k to 10 allows CNN+ANNr to outperform CNN+ANNc, while CNN+3CosMul remains the least performant approach.

7.4.2 Choice of the retrieval metric for retrieval for ANNr

CNN+ANNr is an analogy solving model that requires a retrieval step, where a predicted embedding $\widehat{e(D)}$ is compared with the embeddings of all the candidate words in the vocabulary. The word with the closest embedding is then chosen as a solution, as is done in many other embedding based retrieval approaches. To determine this closest embedding, we minimize a distance or maximize a similarity. The main candidates were Euclidean distance and cosine similarity, and their retrieval performance in preliminary experiments is reported in Table 7.8. Cosine similarity outperforms Euclidean distance in 7 of the 11 languages from Sig16 and JBATS, however the difference is not significant enough to favor either measure.

To summarize, while we use cosine similarity for retrieval with CNN+ANNr, this choice is for the most part arbitrary.

7.4.3 Reconstruction task performance

Siganalogies covers more than 80 languages, of which we exclude low resource languages from Sig19 as well as Basque and Uzbeck, as they contain less than 50000 APs in the training set. What remains is 10 languages from Sig16 and 42 from Sig19, with 6 languages (Arabic, Finnish, German, Russian, Spanish, and Turkish) appearing in both datasets. A language present in both Sig16 and Sig19 is counted as two distinct languages. The AE-emb is trained on these 52 languages, with 5 different random data splits to obtain the training development and test sets, and 10 random initializations per model, for a total of 2600 models ($5 \times 10 \times 52$). In this subsection and in Table 7.9 we detail the accuracy of the models at word level, *i.e.*, if even a single character is mispredicted, the prediction is counted as a failure. For predictions, we use the most likely character at each successive generation step.

The large standard deviation in many languages indicates that, while for some models performance is high, using only the reconstruction task with the words from the data of each language is insufficient to obtain stable performance. This issue transfers to the analogy solving downstream task if no fine-tuning is done, as can be seen in the performance of AE+par. in Table 7.7. It is likely that using a larger lexicon of words will improve the stability and versatility of the AE-emb. This extension was not explored as AE+ANNr results in significantly smaller fluctuations in performance, and designing the best possible embedding model for morphology was not a focus of our work.

For the experiments in Subsections 7.4.1 and 7.4.4, we consider only languages where the AE-emb achieve above 80% reconstruction task accuracy on the first data split. These languages are indicated in bold in Table 7.9. For AE+ANNr and AE+par., only the 10 AE-emb trained on the first data split are used.

Language	Accuracy (%)	Language	Accuracy (%)
Sigmorphon 2016		*Sigmorphon 2019*	
Arabic	75.72±13.98	**French**	76.04±10.20
Finnish	73.30± 9.96	German	64.98±16.12
Georgian	87.06± 6.28	Greek	39.44±21.68
German	69.50±15.03	**Hebrew**	91.16± 7.29
Hungarian	83.57± 6.63	Hindi	58.08±32.85
Maltese	77.07±25.07	Hungarian	61.51±13.81
Navajo	38.36±21.41	Irish	10.39±12.59
Russian	84.70± 6.63	Italian	70.92±13.52
Spanish	78.36±15.85	Kannada	0.05± 0.16
Turkish	83.03±14.07	Kurmanji	76.52± 8.50
Sigmorphon 2019		Latin	69.90±13.80
Adyghe	81.90± 7.35	Latvian	78.29± 5.53
Albanian	51.07±13.56	Persian	59.14±23.10
Arabic	78.40±12.37	Polish	67.36±18.58
Armenian	77.94± 4.87	**Portuguese**	62.88±24.26
Asturian	73.71±26.71	Romanian	62.76±22.14
Bashkir	80.38±18.55	Russian	67.76±11.51
Belarusian	61.15±15.30	**Sanskrit**	83.83± 5.37
Bengali	47.78±25.46	**Slovak**	82.62± 6.66
Bulgarian	76.07± 7.18	**Slovene**	80.95± 8.19
Czech	61.08±19.61	Sorani	77.43±11.88
Danish	73.88±11.70	Spanish	67.86±21.80
Dutch	63.90±19.41	**Swahili**	81.94±21.80
English	65.61±19.96	Turkish	68.03±12.29
Estonian	66.78± 9.02	Urdu	52.02±28.24
Finnish	34.68±22.89	**Welsh**	87.62±12.69
		Zulu	81.96±15.81

Table 7.9: Appendix Table A3 from [MC24]. Accuracy (in %, mean ± std.) at the word level, of the AE pre-trained for at most 100 epochs on 40,000 random words, for 5 different training / test splits and 10 random initialization of the model in each case. Languages in bold are the ones selected for further experiments.

	$\mathcal{L}_{\text{CEL}}$	$\mathcal{L}_{\text{norm. other}}$	$\mathcal{L}_{\text{norm. random}}$	$\mathcal{L}_{\text{all}}$
Arabic (from scratch)	8.53 ± 0.89	16.01 ± 4.20	27.90 ± 4.66	16.45 ± 4.35
Arabic (transfer)	6.86 ± 1.08	68.84 ± 5.54	$\mathbf{73.08 \pm 4.19}$	68.59 ± 3.17
Finnish (from scratch)	21.31 ± 2.31	64.14 ± 7.07	47.25 ± 9.82	57.35 ± 5.46
Finnish (transfer)	6.23 ± 1.27	87.37 ± 2.11	$\mathbf{90.15 \pm 1.42}$	89.90 ± 1.61
Georgian (from scratch)	34.14 ± 3.44	76.33 ± 7.92	68.56 ± 9.85	77.66 ± 5.42
Georgian (transfer)	16.82 ± 3.33	95.64 ± 1.12	$\mathbf{97.12 \pm 0.26}$	95.09 ± 0.65
German (from scratch)	29.31 ± 3.42	64.05 ± 11.12	51.95 ± 11.21	54.91 ± 8.81
German (transfer)	14.17 ± 2.02	90.66 ± 0.66	91.47 ± 0.55	$\mathbf{91.52 \pm 0.62}$
Hungarian (from scratch)	24.35 ± 1.64	50.11 ± 4.72	41.25 ± 9.39	55.54 ± 4.46
Hungarian (transfer)	11.61 ± 1.93	$\mathbf{89.85 \pm 1.52}$	89.42 ± 1.43	88.79 ± 1.78
Maltese (from scratch)	5.70 ± 0.88	48.21 ± 9.24	72.74 ± 3.52	51.82 ± 10.22
Maltese (transfer)	5.63 ± 0.84	95.94 ± 0.73	$\mathbf{97.16 \pm 0.30}$	91.69 ± 1.55
Navajo (from scratch)	6.81 ± 0.69	19.36 ± 3.07	23.66 ± 2.99	24.69 ± 2.49
Navajo (transfer)	6.34 ± 0.64	$\mathbf{53.73 \pm 1.58}$	52.22 ± 1.18	52.45 ± 1.97
Russian (from scratch)	12.21 ± 1.72	38.46 ± 4.77	36.80 ± 3.84	41.04 ± 4.00
Russian (transfer)	4.47 ± 0.68	74.08 ± 1.24	71.66 ± 0.76	$\mathbf{76.12 \pm 1.11}$
Spanish (from scratch)	26.76 ± 1.70	84.26 ± 2.93	78.31 ± 5.91	81.76 ± 4.19
Spanish (transfer)	20.67 ± 4.32	91.03 ± 1.63	$\mathbf{92.63 \pm 1.25}$	91.12 ± 1.65
Turkish (from scratch)	13.30 ± 1.07	31.71 ± 4.40	42.82 ± 5.49	39.16 ± 6.82
Turkish (transfer)	7.41 ± 1.63	86.17 ± 1.66	88.36 ± 1.17	$\mathbf{88.45 \pm 1.12}$
Japanese (from scratch)	23.33 ± 2.71	9.76 ± 2.52	34.00 ± 8.54	11.46 ± 3.54
Japanese (transfer)	26.89 ± 1.66	83.10 ± 0.92	$\mathbf{86.12 \pm 0.76}$	74.14 ± 1.09

Table 7.10: Table 1 from [Mar+22a]. Top-1 accuracy (*i.e.*, precision, in %) of ANNr when training for 50 epochs with the embedding from ANNc and from scratch. We report mean ± std. over 10 random seeds. Boldface results are the best for each language.

7.4.4 Training procedure for ANNr

In [Mar+22a], we compared $\mathcal{L}_{\text{CEL}}$, $\mathcal{L}_{\text{norm. other}}$, $\mathcal{L}_{\text{norm. random}}$, and their combination $\mathcal{L}_{\text{all}}$, to determine which criterion was more suitable for CNN+ANNr. We also confirmed that fine-tuning the CNN-emb trained with ANNc improved analogy solving performance by a wide margin, compared to training CNN-emb with ANNr from scratch. The experiments were done on Sig16 and JBATS, and each model is trained for 10 random initializations. The results are reported in Table 7.10.

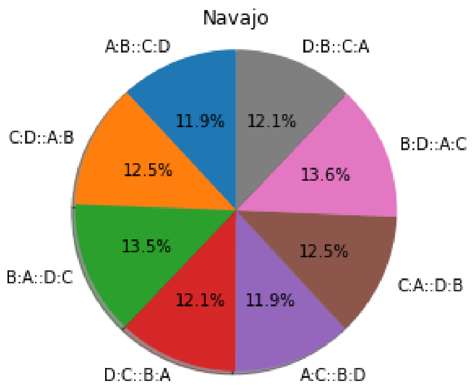
Pre-training the embedding for ANNr. Reusing the embedding model trained with ANNc reduces the convergence time, which is expected when performing transfer learning between closely related tasks. What is more interesting is that fine-tuning a pre-trained CNN-emb significantly improves the final performance. In contrast, when training CNN-emb from scratch, we observe much lower performance with all criteria. Removing the limit of 50 training epochs in further experiments did not significantly improve those results. This indicates that ANNr is not enough to learn CNN-emb from a random initialization, while ANNc seems suitable to pre-train CNN-emb on analogy detection.

Training losses for ANNr. A first observation is that $\mathcal{L}_{\text{CEL}}$ is, by a significant margin, the worst performing criterion in all settings. With further experiments, we confirmed poor performance with both Euclidean distance and cosine similarity to retrieve the solution to the analogical equation. This can appear surprising, as the CEL optimizes the cosine similarity, and the latter should be particularly compatible with a model trained using the former.

In most cases when using the pre-trained CNN-emb, using $\mathcal{L}_{\text{norm. random}}$ performs better than or comparably to $\mathcal{L}_{\text{norm. other}}$ and $\mathcal{L}_{\text{all}}$. $\mathcal{L}_{\text{all}}$ only performs better than the other criteria on Russian. Based on these observations, we decided to use $\mathcal{L}_{\text{norm. random}}$ in our later experiments, as it is the one that requires the least computations and training is slightly faster.

7.4.5 Case study on Navajo and Georgian

To get further insight on the behavior of CNN+ANNr, we performed a more detailed analysis of where the model makes mistakes for two languages: Navajo and Georgian [Mar+22a]. These experiments were performed before we started using Sig19, and used data from Sig16 and JBATS. Out of these languages, CNN+ANNr performed the best on Georgian (accuracy of 97.12 ± 0.26) and the worst on Navajo (52.22 ± 1.18). For each language, we consider only one of the 10 random initialization. The results reported in Table 7.7 are from experiments recomputed at a later time, which explains why we do not have exactly the same performance for Georgian.



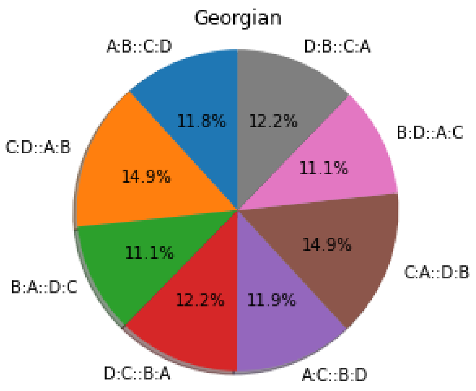


Figure 7.3: Rate of appearance of each permutation of the base APs in the errors made by CNN+ANNr. The sum of percentages in each pie chart actually makes 100%, inconsistencies are caused by rounding.

Permutations involved in errors. For both languages, we analyse how frequently each of the 8 equivalent forms cause a mistake. The rate of appearance of each permutation among all mistakes is presented in Figure 7.3.

There is no significant predominance of a particular permutation in the mistakes on either Navajo or Georgian. This shows that ANNr is, to some extent, invariant to the permutation postulates used in the data augmentation process.

For easier comparison, permutations with similar appearance rate were put in front of one another. Notice that en each case, pairs of permutations that have the same fourth element also have the same appearance rate among errors. This hints that for CNN+ANNr, Exchange of the Means does not change the difficulty of an analogical equation, which is consistent with the intuition of the architecture of ANNr.

Errors when stronger versions of Identity or Reflexivity of Conformity are not respected by the data. Less than 0.3% of mistakes on Navajo involve expected cases of Identity or Reflexivity of Conformity, *i.e.*, forms like $A : B :: A : x, x = B$ or $A : A :: B : x, x = B$, and no such case was observed for Georgian. This indicates that Reflexivity of Conformity is well handled by CNN+ANNr, in particular since Reflexivity of Conformity is not true by definition for CNN+ANNr, contrary to, for instance, the parallelogram rule.

By contrast, we notice that 47.49% of all mistakes in Georgian are cases where either of the following happens in terms of word forms.

1. $A = B$ but $C \neq D$ and the model predicts $\widehat{D} = C$ or, similarly by Exchange of the Means, $A = C$ but $B \neq D$ and the model predicts $\widehat{D} = B$ (41.06% of all mistakes). In English, a similar situation would be "*sing*" : "*sing*" :: "*am*" : x, in which expecting $x =$ "*are*" is counter-intuitive if the underlying transformation from the first to the second "*sing*" ("pos=V, per=1, num=SG" to "pos=V, per=1, num=PL") is not provided.

2. $A \neq B$ but $C = D$ and the model predicts $\widehat{D} \neq C$, or $A \neq C$ but $B = D$ and the model predicts $\widehat{D} \neq B$ (6.42% of all mistakes). Using the same example as before, if not enough is known of the language, "*am*" : "*are*" :: "*sing*" : $x, x =$ "*sing*" can also appear a bit counter intuitive.

It is interesting to note that the model makes more mistakes when presented with an apparent instance of Identity ("*sing*" : "*sing*" :: "*am*" : x) than when the expected answer is the same as one of the elements ("*am*" : "*are*" :: "*sing*" : x). To the best of our knowledge, this corresponds to the behavior of humans, who are able to transfer from a complex case to a simpler one, but would prefer a simpler relation over a complex one [Mur+20]: identity is simpler that any morphological transformation. This also matches the data augmentation process, where $A : A :: B : C$ is invalid while $A : B :: C : C$ is not generated, at least if we exclude cases such as "*sing*" : "*sing*" from the previous examples.

The permutations described in case 1 above correspond to violations of stronger versions of Reflexivity of Conformity, introduced in [Lep03] as Strong Identity ($A : A :: B : x \implies x = B$) and Strong Reflexivity of Conformity ($A : B :: A : x \implies x = B$). ANNr appears to implicitly learn these two properties, that are not enforced by the data augmentation. Besides, we can see in Tables A.1 and A.2 (Appendix B) that the expected answer is usually very close to the predicted $\widehat{e(D)}$. This includes cases where the data does not respect Strong Identity or Strong Reflexivity of Conformity.

Complex morphology of Navajo verbs. Verbs correspond to 73.79% of all mistakes in Navajo, or 75.35% if we exclude cases described in the previous paragraph. As Eddington and Lachler state:

> "Verb stem inflectional patterns in Navajo are arguably one of the most intractable problems in modern Athabaskan linguist studies." [EL10]

They also refer to the work of Leer which states that this complexity can be attributed to "analogical innovation, which is thus quite difficult to analyze synchronically" [Lee79]. This could explain why all approaches, including the symbolic baselines, have a low performance on Navajo. Despite that, CNN+ANNr reaches above 92% of hit rate at 10.

7.5 Sensitivity of the model to input perturbations

Epistemic uncertainty [Gal16], also called model uncertainty, refers to uncertainty in model parameters and structure. For instance, "a large number of possible models might be able to explain a given dataset, in which case we might be uncertain which model parameters to choose to predict with." In [Als+21c], we performed experiments to evaluate the epistemic uncertainty of CNN+ANNc and CNN+ANNr, and determine if structural changes might benefit performance. We tested their sensibility to perturbations of the input by applying random dropout on the embeddings during evaluation.

Input perturbation using dropout In DL, applying a random dropout with a probability p_{dropout} on a set of values (for instance, a vector) means that each value (each component of the vector) has a probability p_d to be replaced by, for instance, 0. We introduce such a dropout on the embedding produced by CNN-emb. Therefore, some meaningful information contained in the embedding is radomly lost, which perturbates the input of CNN+ANNc and CNN+ANNr.

We evaluate the performance of the models with several dropout probabilities: 0.01, 0.05, 0.1, 0.3 for analogy detection, and for analogy solving 0.005 and 0.5 are also considered. For each language, a single model initialization for CNN+ANNc and 3 initialization for CNN+ANNr were considered. For each dropout value and each model initialization, the experiment is repeated 10 times to account for the stochastic nature of dropout.

Results and discussion. The results are summarized in Figures 7.4 and 7.5, with the error bars representing the standard deviation. Note that the stochastic aspect of dropout does not bring much variance in the results on CNN+ANNc, as the significant number of test examples absorbs the randomness of dropout. We still observe a notable variance for CNN+ANNr, but it is almost exclusively inter-model variance between the 3 models trained. For each model, the results with each dropout probability is significantly different (Student t-test $p < 0.01$, details in Table 7.11) from adjacent probabilities.

For both the regression and classification models, we observe that the more perturbed the input is, the lower the performance. As our models use the information in the embeddings, this behavior is expected. What is more interesting is that there is a large drop in performance past a certain

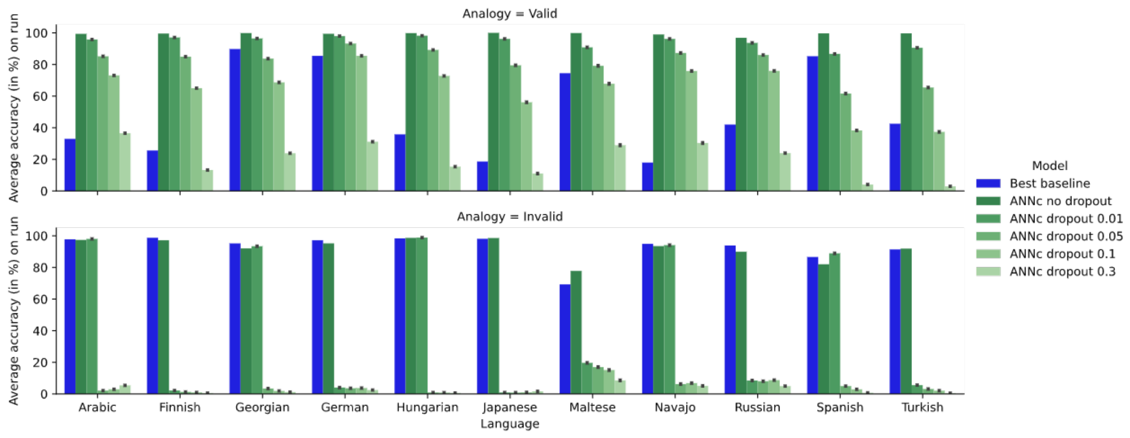


Figure 7.4: Figure 2 from [Als+21c]. Classification accuracy (in %) for each language with input perturbation using various dropout probabilities. Error bars correspond to standard deviation.

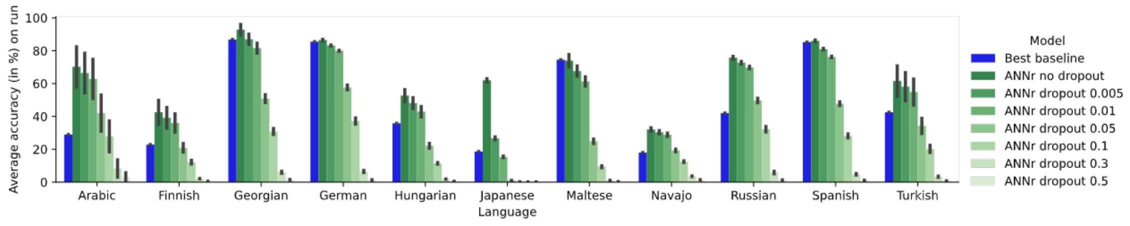


Figure 7.5: Figure 3 from [Als+21c]. Regression accuracy (in %) for each language with input perturbation using various dropout thresholds. For each of the 3 models of each language, the results with each dropout probability is significantly different from the other probabilities (Student t-test $p < 0.01$). Error bars correspond to standard deviation.

threshold, in particular for CNN+ANNc on invalid APs. For example, for Arabic and Hungarian the threshold is between $p_d = 0.05$ and $p_d = 0.1$, while for Maltese and German it is between $p_d = 0.01$ and $p_d = 0.05$. This effect is also present for analogy detection on valid APs and on analogy solving, even if more subtle. Note that the threshold differs between languages, between analogy detection and analogy solving, and for each language between valid and invalid APs.

Taking analogy detection on invalid APs in Arabic as an example, this drop in performance can be interpreted as the model managing to perform well even when 1% of the embedding is missing. In other words, those 1% are likely to contain redundant or unnecessary information, as dropping them has little impact. However, once we remove 5%, the model performance crashes, so these 5% likely contain necessary information. In practical terms, we could most likely reduce the embedding size by 1% for Arabic without affecting the classification performance much, considering the negligible drop in performance for valid AP at 1% dropout.

For analogy solving, some languages do not show the plateau preceding this characteristic drop in performance, in particular Japanese. By extrapolation, we could expect the performance to increase for those languages with larger embedding models.

		0.05	0.1	0.3
+	0.01	0.00045	5.00110×10^{-6}	6.44357×10^{-15}
	0.05	/	0.00765	1.56664×10^{-11}
	0.1	/	/	1.74502×10^{-7}
−	0.01	0.00701	0.00653	0.00512
	0.05	/	0.83337	0.28144
	0.1	/	/	0.38175

Table 7.11: Appendix Tables 8 and 9 from [Als+21c]. Independent t-test p-value on the evaluation of valid (+) and invalid (−) APs with different dropout settings, for CNN+ANNc.

7.6 Postulates for CNN+ANNc: case of Exchange of the Means

We describe in this section experiments from [MMC22], in which model transfer is used to confirm that changing the postulates used in the data augmentation process results in a sigificantly different model matching the changed axiomatic settings. The experiments focus on Exchange of the Means, that is not suitable for some application settings [Als+22b; Ant22], as discussed in Subsection 2.3.2.

Experimental setting. In this experiment we consider the three settings EM, $\neg EM$, and $\overline{EM}$ defined in Subsection 6.3.2. We use them as training settings and test settings for all 9 possible combinations, represented using the "training setting → test setting" notation. The three settings correspond to the default axiomatic setting of APs (EM) as well as two variants based on how Exchange of the Means is considered:

- EM (using P^+_{EM} and P^-_{EM}), where Exchange of the Means is accepted;
- $\neg EM$ (using $P^+_{\neg EM}$ and $P^-_{\neg EM}$), where Exchange of the Means is not considered;
- $\overline{EM}$ (using $P^+_{\overline{EM}}$ and $P^-_{\overline{EM}}$), where Exchange of the Means is specifically invalid.

Experiments were done with CNN+ANNc, as it uses invalid permutations for training and testing. For training, we use the (up-)sampling described in Subsection 7.3.3 to obtain 8 valid and 8 invalid permutations, and consider 10 random initializations. The experiment is repeated on 11 languages of Sig16 and JBATS. All other hyperparameters follow the default values from Subsection 7.1.2.

Expected performance. Intuitively, the expected behavior is the following, also represented in the top left corner of Figure 7.6 and forming a charracteristic Z shape:

- each model is expected to perform best on when the training setting is the same as the test setting ($EM \rightarrow EM$, $\neg EM \rightarrow \neg EM$, and $\overline{EM} \rightarrow \overline{EM}$);
- both $EM \rightarrow \overline{EM}$ and $\overline{EM} \rightarrow EM$ are expected to perform poorly, as the source and target settings are incompatible with regards to Exchange of the Means;
- both EM and $\overline{EM}$ are expected to perform well on $\neg EM$, as the permutations in $\neg EM$ are common to both EM and $\overline{EM}$;
- the performance for $\neg EM \rightarrow EM$ and $\neg EM \rightarrow \overline{EM}$ are hard to predict, as $\neg EM$ is a subset of both EM and $\overline{EM}$.

Results and Discussion In Figure 7.6 we report the balanced accuracy. For all languages we observe the expected results with minor variations:

- first, on the test setting $\neg EM$, all the models perform equally, instead of $\neg EM \rightarrow \neg EM$ performing slightly better;
- second, the performance of $EM \rightarrow \overline{EM}$ and $\overline{EM} \rightarrow EM$ are not as low as expected, with the peculiarity that $EM \rightarrow \overline{EM}$ always outperforms $\overline{EM} \rightarrow EM$ by roughly 10%.

These results confirm that the training procedure does have an impact on which permutations will be considered valid or invalid by the model. Furthermore, the observed results match the expected Z shape for all languages, which supports the intuitions used to construct the expected results. As these intuitions rely on the differences between the axiomatic settings, this experiment confirms that using specific postulates to train a model results in models with properties fitting the corresponding axiomatic setting. Additionally, the model is trained on data and is able to handle postulates that are not explicit in the data augmentation process, like the Strong Identity and Strong Reflexivity of Conformity mentioned in Subsection 7.4.5.

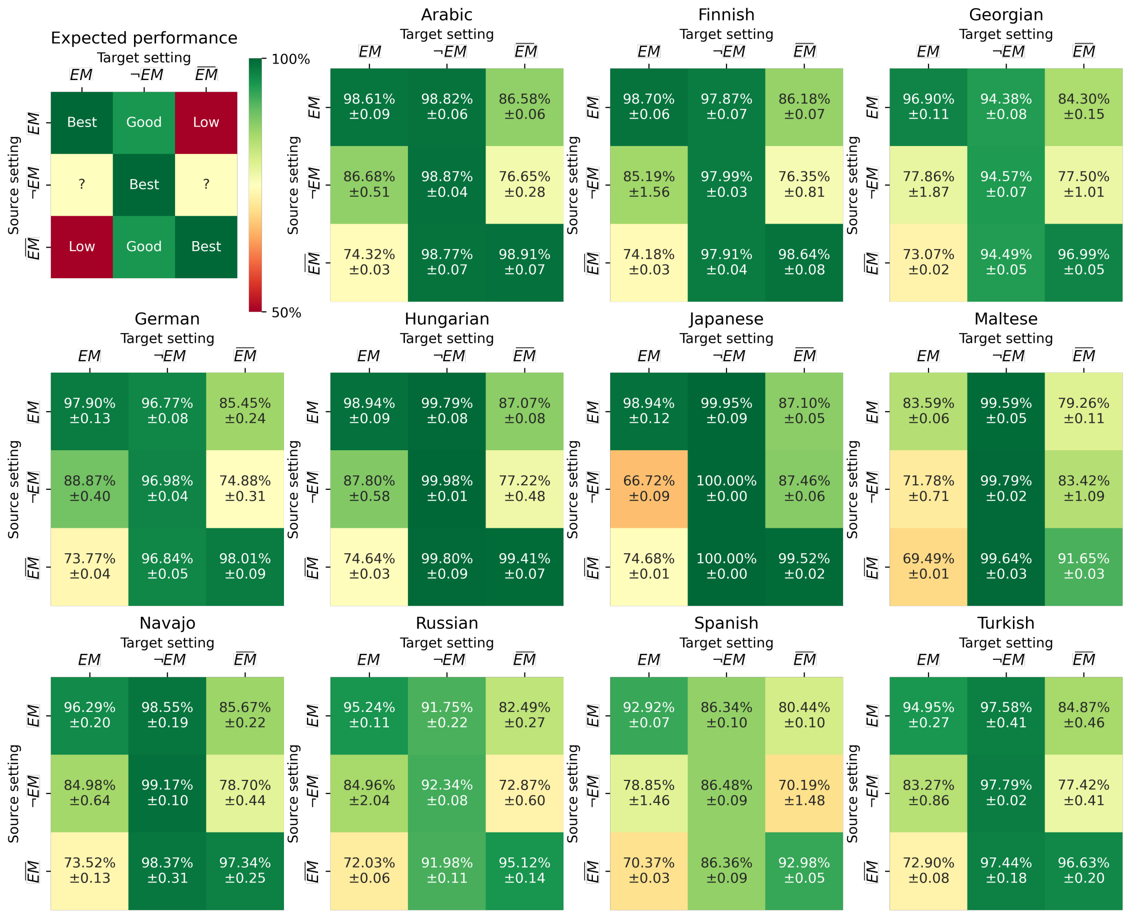


Figure 7.6: Figure 1 from [MMC22].Balanced accuracy of CNN+ANNc, 10 per training setting. In the top left corner, a representation of the expected results. Source setting corresponds to the training setting, and target setting corresponds to the testing setting.

7.7 Discussion and perspectives

The ANN framework we propose for analogy detection and analogy solving uses ANNc and ANNr, two DL models inspired from the properties of APs. The ANN framework outperforms non-parametric approaches and symbolic baselines on morphology, and outperforms other approaches approaches on semantics, as discussed in Subsection 7.7.1. A significant part of the performance of the approach comes from the analogical data augmentation and some aspects of the training process, which are discussed in Subsection 7.7.2. Some results when using dropout on the word embeddings indicate that performance could be improved by refining the structural hyperparameters, as discussed in Subsection 7.7.3.

The ANN framework we study offers multiple ways to tackle analogy detection and analogy solving, and we discuss the main pros and cons of each method in Subsection 7.7.4. Despite the alternatives we propose, it can be argued that the architecture of ANNc and ANNr are not the most fitting for the manipulation of APs, yet they are relatively easy to grasp intuitively and achieve good performance. Perspectives to integrate other models to the ANN framework are discussed in Section 6.5.

In our experiments, we put to light some known and less known limitations of the symbolic approaches we use as baseline. These limitations are discussed in Subsection 7.7.5.

7.7.1 ANNc and ANNr outperform symbolic, non-parametric, and other parametric approaches.

Non-prametric models of APs. As shown in [MC24] and in the experiments of this Chapter 7, the performance achieved by the ANN framework is higher than the one of non-parametric models such as 3CosMul or the parallelogram rule on analogy solving.

Symbolic approaches to morphological APs. Our framework also outperforms symbolic approaches to morphological APs. In all our works, we compare ourselves with Alea [LYZ09] on Kolmo [Mur+20] on analogy detection and analogy solving, as well as with Nlg [FL18] in most cases on analogy detection.

In Section 7.3, we compare CNN+ANNc with Nlg, as well as adaptations of Alea and Kolmo to analogy detection. It can be argued that comparing generative approaches to analogy solving (Alea and Kolmo) with analogy detection approaches is unfair, and this was taken into account before drawing conclusions. Additionally, up to 10 generated solutions were considered for valid and invalid APs, which results in higher accuracy, in particular for valid APs. In this setting, we showed that ANNc still outperforms the baselines on valid APs with no significant difference on invalid APs.

In Subsection 7.4.1, CNN+ANNr and AE+ANNr outperform the symbolic baselines on analogy solving by a very significant margin in all cases. Furthermore, by considering up to 10 retrieval results for CNN+ANNr (see Appendix Tables A.1 and A.2), the accuracy improves above 99% for all but 6 languages which reach above 95%. This result is significant, as 10 candidates is a small fraction of the candidate words: it ranges from less than 0.16% of all candidates (6419) for Swahili to less than 0.04% for Georgian (32233 candidates).

Parametric models. For semantic APs between words, Lim, Prade, and Richard [LPR21] showed the benefit of the ANNc and ANNr models over MLP (see Subsection 3.2.1) up to 5 layers, random forest [Ho95] and support vector machine [CV95]. The performance improvement is especially striking for ANNr, while for ANNc the benefits are seen mostly on the SAT-based task which is considered a harder semantic analogy task [LPR21].

7.7.2 Discussion on the training process and analogical data augmentation

To achieve the best possible performance, we experimented with many different parameters of the training procedure to reach the current state of the framework. In particular, we refined the analogical data augmentation process to eliminate class imbalance when training ANNc and embedding collapse when training ANNr, and explore different axiomatic settings for analogies. It would be beneficial to further refine our training process as well as the training parameters and hyperparameters, but we leave those considerations for further work due to time limitations.

In the results on morphology and in applications of the analogical data augmentation process to different domains [Als+22b; Zer+22, see also Chapter 10], we observe that analogical data augmentation reduces the dependence of the model on its initialization and on minor variations in model input, as discussed in the last paragraph of this subsection.

Evolution of the data augmentation process for ANNc. Regarding ANNc, in [Als+21a] we identified an imbalance in the analogy detection performance due to the data augmentation process, as we later confirmed in [Als+21c]. Using the 8 equivalent forms of a valid AP together with the 24 corresponding invalid forms skewed the data in favor of invalid analogies, while using only the 3 invalid forms from the base AP skewed the data in the other direction, resulting in a matching imbalance in the performance over the two classes for CNN+ANNc. A compromise was found by sampling 8 out of the 24 invalid forms, presented in Subsection 6.3.3 and demonstrated in Subsection 7.3.3. Based on detailed analyses of results of some languages, in particular from [Mar+22a; MMC22], we further refined the data augmentation to exclude invalid forms that also appear in the valid forms.

Evolution of the training process for ANNr. The current training procedure of ANNr is the result of multiple observations along our experiments reported in Subsections 7.4.2 and 7.4.4.

Indeed, pre-training the embedding is necessary to achieve good performance on analogy solving with CNN+ANNr, and this can be achieved with relative ease by using CNN+ANNc, with a 10% to 40% improvement reported in [Mar+22a, see also Table 7.10], or using an reconstruction task as we identified in Subsection 7.4.1. In preliminary experiments, we also observed that training only the ANNr part of the model resulted in poor performance, and training CNN+ANNr with a naive MSE resulted in a collapse of the embedding space. After experimenting with various training criterion based on these results, we achieved the best results with $\mathcal{L}_{norm.random}$. While $\mathcal{L}_{norm.random}$ is an easy to implement way to mitigate embedding space collapse, in many settings it achieves comparable performance to the other training criterion we considered, so we encourage future users of ANN framework to also consider the other training criterion.

Behavior of the models with regards to the axiomatic setting. In [MMC22, see also Section 7.6], multiple variants of the axiomatic setting of APs are considered for transfer experiments. These variants cover different ways to consider the postulate of Exchange of the Means: the way it is considered with APs (*i.e.*, as an accepted postulate), a variant where it is not an accepted postulate, and a last variant where it is actively considered as a property quadruples should not have. The transfer experiments between the three settings confirmed that changing the postulates used to define the data augmentation process result in significantly different models, corresponding to the changed axiomatic settings.

These results highlight the importance of the choice of an axiomatic setting fitting the use case when using analogical data augmentation, in particular since some postulates are discussed for some application settings (see Subsection 2.3.2). Some guidelines and considerations are discussed in Appendix D, in which we explore different combinations of postulates and the resulting data augmentation processes. However, experiments with the various subsets of postulates are required to confirm the proposed generalized data augmentation procedure.

If enough analogical data is available, we hypothesize that it is possible to determine the most fitting set of postulates by finding the axiomatic setting with the closest performance to the non-augmented analogical data, but experiments on a wider range of application domains would be required to confirm this hypothesis. When analyzing the results of Navajo and Georgian in [Mar+22a, see also Subsection 7.4.5], we identified several interesting behaviors of CNN+ANNr: *(i)* there is no significant performance difference based on permutation, which indicates that the model is invariant to the postulates of APs used in training, *(ii)* Reflexivity of Conformity and Identity ($A : B :: A : B$ and $A : A :: B : B$ respectively) are well handled, *(iii)* when an example appears to violate Strong Reflexivity of Conformity or Strong Identity (*e.g.*, "*ran*" : "*ran*" :: "*was*" : x, $x =$ "*were*") the CNN+ANNr model frequently gives an answer different from what is expected. Te given answer corresponds to a proper application of the violated postulate.

Reduced dependence on initialization and minor variations in model input. The results in [MC24, see also Section 7.4] indicate that data augmentation reduces the sensitivity to the initial setting of the model, with significantly smaller standard deviation after fine-tuning using ANNr. Additionally, in [Zer+22, see also Chapter 10], we applied the data augmentation process and ANNc to TSV, and observed that using ANNc together with our data augmentation during training reduces the sensitivity of the model to some variations in the input encoding. Overall, it appears that models trained with the analogical data augmentation are less sensitive to slight changes in their input, as they are made invariant to changes in the input due to the postulates of APs.

7.7.3 Perturbation of model inputs and embedding dimension

We explored in Section 7.5 [Als+21c] the sensitivity of CNN+ANNc and CNN+ANNr to perturbations in their input, by applying dropout on the word embeddings. By randomly replacing some embedding components by 0 with a given probability, we found that beyond a certain dropout probability the performance of CNN+ANNc dropped drasticaly for invalid APs, and observed a similar but less striking phenomenon for CNN+ANNc on valid APs and for CNN+ANNr. The probability threshold depends on the language we consider, and we observe a plateau of high performance before the threshold. This allowed us to identify languages for which the embedding appears to contain redundant information (hence there is little effect of removing a few) or conversely languages that might benefit from an increase in embedding size (where the plateau of high performance does not appear, and that are more sensitive to dropout). In the latter category we find Japanese, which is in line with the larger set of possible characters.

7.7.4 Diversity in tackling APs and application to other domains

Our framework proposes multiple technical solutions, in particular for analogy solving, each with their own benefits. For instance, while CNN+ANNr outperforms CNN+3CosMul, the latter does not require training an additional model. Also, while using CNN+ANNc for retrieval outperforms CNN+3CosMul without requiring additional training either, we do not recommend using the former due to two key limitations: *(i)* the execution time for retrieval is greater than the time needed to an CNN+ANNr model or to use traditional cosine-based retrieval, and *(ii)* CNN+ANNc tends to give similarly high scores to multiple solutions, which entails the expected solution appearing further in the ranking. However, *(ii)* can be a desired property of the model, and *(i)* can be mitigated by engineering the prediction process, for instance by preprocessing the candidates more effectively.

Additionally, while we use cosine similarity for retrieval with CNN+ANNr based on preliminary experiments, the difference between cosine similarity and Euclidean distance as shown in Table 7.8 was not necessary significant, so it can be relevant to test both approaches for other applications.

Beyond the choice of the model, using ANN framework requires to formulate the problem to tackle as APs or analogical equations. This step can be challenging, as can be seen in Chapter 11, or in [Als+22b] or Chapter 10 where multiple formulations are explored. Nevertheless, the effort is usually rewarded by good performance and the integration of analogical knowledge in the model.

7.7.5 Some limitations of symbolic approaches

Along our experiments, and in particular in [Mar+22a], we identified several limitations of the Alea and Kolmo symbolic approaches that we use as baselines in most of our experiments. For instance, longer words and words with many repeated adjacent letters significantly reduce the speed of the two approaches, with this phenomenon particularly striking for Kolmo, as was already identified by the authors of [Mur+20]. We solved this issue by introducing a timeout for Alea and Kolmo as described in Subsection 7.2.4, as it appears that for most languages less than 0.5% of the analogical equations took longer than 10 seconds to solve.

Additionally, in [Als+21a] we found that Kolmo struggled with APs obtained in the data augmentation by using Exchange of the Means. It is interesting that this approach, based on empirical observations of human behavior in analogy solving, does not follow all the postulates of APs. This highlights the need for flexibility with regards to the setting of APs. This flexibility can be provided by the ANNc and ANNr models, and is one of the main advantages of the ANN framework over symbolic and non-parametric approaches.

Chapter 8

Cross-lingual transfer and multilingual models

Chapter contents

In this chapter, we present experiments on modeling multiple languages at once and transferring models between languages with CNN+ANNc [Als+21a; Als+21b; MMC22]. We discuss the findings from our experiments on cross-lingual transfer and multilingual models, in Section 8.3.

In [Als+21b], we experimented with different forms of transfer from a source language to a target language without fine-tuning, with languages from Sig16 and JBATS. We also experimented with different approaches to model multiple languages at one with a single CNN+ANNc. These experiments are reported in Section 8.1.

As reported in Section 8.2, we performed further transfer experiments in [MMC22], by transferring CNN+ANNc between languages from Sig19, which covers a wider range of languages.

8.1 Multilingual experiments on Sig16 and JBATS

We performed in [Als+21b] two sets of multilingual experiments, reported in this section. A first group of experiments was performed to determine how models trained on one language perform on other languages, reported in Subsection 8.1.1. The second group of experiments, described in Subsection 8.1.2, explored the potential of training on data from multiple languages to obtain a model of morphological APs that generalizes better to the morphology of multiple languages.

Following this idea, we trained several CNN+ANNc models and evaluated how they perform across the 11 languages of Sig16 and JBATS. At the time of the experiment, we already identified that the class imbalance in the 8/24 setting with 8 valid and 24 invalid permutation, used in [LPR21], causes imbalance in performance, and that the 8/3 setting performs slightly better (see Subsection 7.3.3 for the definition of the imbalance settings). We also did not yet implement the balanced 8/8 setting with sampling (see Subsection 7.3.3) which would have probably given better results. Thus, training was done in the 8/3 setting with 8 valid and 3 invalid samples. This limitation is addressed in Section 8.2, where the 8/8 setting is used for training.

8.1.1 Transfer performance in Sig16 and JBATS

As mentioned above, our first experiments focussed on transferring models from one language to the another language. In practice, this corresponds to training CNN+ANNc on one language and testing it on another language.

Two variants of the transfer metodology. Accounting for the gaps in the morphology of different languages, we devised two variants of the transfer methodology from a language L_{training} to a L_{test} language:

- in full transfer, both the CNN-emb and ANNc models trained on L_{training} are applied on L_{test};
- in partial transfer, only the ANNc part of CNN+ANNc trained on L_{training} is applied on L_{test}, however we use the CNN-emb trained on L_{test}.

The two approaches have different weaknesses as we do not fine-tune the models after transfer. In full transfer CNN-emb has not seen the characters used in L_{test}, and is not trained to handle the morphology of L_{test}. Partial transfer was used to handle this issue, as the CNN-emb is trained on L_{test}, however the embedding spaces of CNN-emb and ANNc might not have the same arrangement.

Results using full transfer. The results for full transfer are reported as heatmaps in Figure 8.1. The results for valid permutations are above 90% except when transferring to Arabic and Navajo, in particular from Finnish, Hungarian, and Spanish.

The results on invalid permutations were more heterogeneous. We observe three languages where performance is poor when transferring from or to them: Georgian, Japanese, and Russian. This is likely due to the different writing systems used by these three languages (compared to the other languages in the dataset), as mentioned in Subsection 5.5.1, and a large portion of the characters are not recognized by the CNN-emb during testing.

Surprisingly, transferring to Japanese does not systematically result in a performance close to 0% on invalid APs. None of the Japanese characters are present in the data of other languages in Sig16, and every word should appear as the empty word ε, causing even invalid APs to look like $\varepsilon : \varepsilon :: \varepsilon : \varepsilon$ and be classified as valid. Following the results from [Lep17], the length of a word can be used to detect some invalid APs, and it is likely that CNN-emb encodes this information for some languages. Indeed, CNN-emb is aware of the distance between the beginning and end of the word, marked respectively by BOW and EOW, within the limit of 6 characters, the largest window size used in CNN-emb.

Other languages have an accuracy between 50% to 80%. Interestingly, models trained on Turkish and Hungarian perform slightly better when compared to those trained on other languages. These results have been used for bilingual models in subsequent experiments reported in Subsection 8.1.2.

Results using partial transfer. As we saw above, not being able to handle the characters used in a language is a huge limitation for the transfer of CNN+ANNc, and partial transfer appears to overcome this limitation from the results reported in Figure 8.2.

The partial transfer performance is high for valid permutations, and lower for invalid permutations, similarly to full transfer. However, we do not observe the distinct lines of close to 0% performance anymore, except for the Georgian to Japanese and Spanish to Japanese transfers. The Spanish to Arabic transfer is at a low 8% and Spanish to Finnish at 16%. Excluding those four cases, performance is above 25% on invalid APs which is a clear improvement from full transfer. However, excluding Georgian, Japanese, and Russian, performance on invalid APs was overall higher in full transfer.

Similarly to full transfer, Hungarian and Turkish are the best performing training languages. While there is no language that appear incompatible with other in partial transfer, we notice that Arabic, Georgian, Japanese, and Spanish have relatively poor performance as training languages. For partial transfer, the alphabet gap is no longer an issue which results in a significant improvement in the overall performance. Nevertheless, since the CNN-emb is not trained together with the ANNc, a mismatch in the representation is likely to appear. Our hypothesis is that this mismatch is part of the cause of the lower performance for Arabic, Georgian, Japanese, and Spanish.

General observations. We distinguish results on valid and invalid permutations of the APs. Results on the base permutations have also been computed. However, as they follow the same trends as valid permutations, the corresponding results have been omitted.

Overall, models perform well when they are applied on the language they were trained on. Performance is high on valid APs including the base forms, but lower on invalid permutations. This bias is consistent with the training 8/3 setting which is imbalanced in favor of the valid APs. Moreover, performance is heterogeneous on valid permutations while there are significant differences based on the training and test language on invalid permutations.

Another interesting observation is that there is no symmetry in the heatmaps, which indicates that the ease of transfer from and to a language are not equivalent.

The high performance of Hungarian as a training language in all settings could be explained by its "particularly rich morphology" [Kie10] using inflection, derivation, and compounding.

8.1.2 Training CNN+ANNc models on multiple languages

Following our first experiments on transferring models between languages in [Als+21b], reported in Subsection 8.1.1, we experimented with multilingual models. The initial intuition was that a model trained on multiple languages would generalize better to new languages, like someone used to learning foreign languages would have an easier time when faced with a new language. Following this idea, we trained several CNN+ANNc models and evaluated how they perform across the 11 languages of Sig16 and JBATS. We explored two different settings regarding the languages we used: in the bilingual setting, we trained models with a subset of two languages as training data and in the omnilingual setting, we trained models with all languages.

For comparable results with ANNc trained on a single language, the amount of training data per language is reduced in proportion with the number of languages used, such that the total remains at 50000 base APs: 25000 APs per language for the bilingual models, and 5000 APs per language for the omnilingual models, except when Japanese is included in which case it is around 4545 APs per language.

As mentioned in Subsection 7.1.2, we use a character embedding size $m = 64$ for the CNN-emb, except for Japanese, where $m = 512$ is used due to a larger amount of distinct characters.

Variants of bilingual models. For our bilingual models, we worked with two pairs of languages:

- Hungarian-Finnish: these two languages are close in terms of "language families" from Wikipedia (see Figure 8.11);
- Hungarian-Turkish: the two languages that produced the best results in terms of transferability in Subsection 8.1.1.

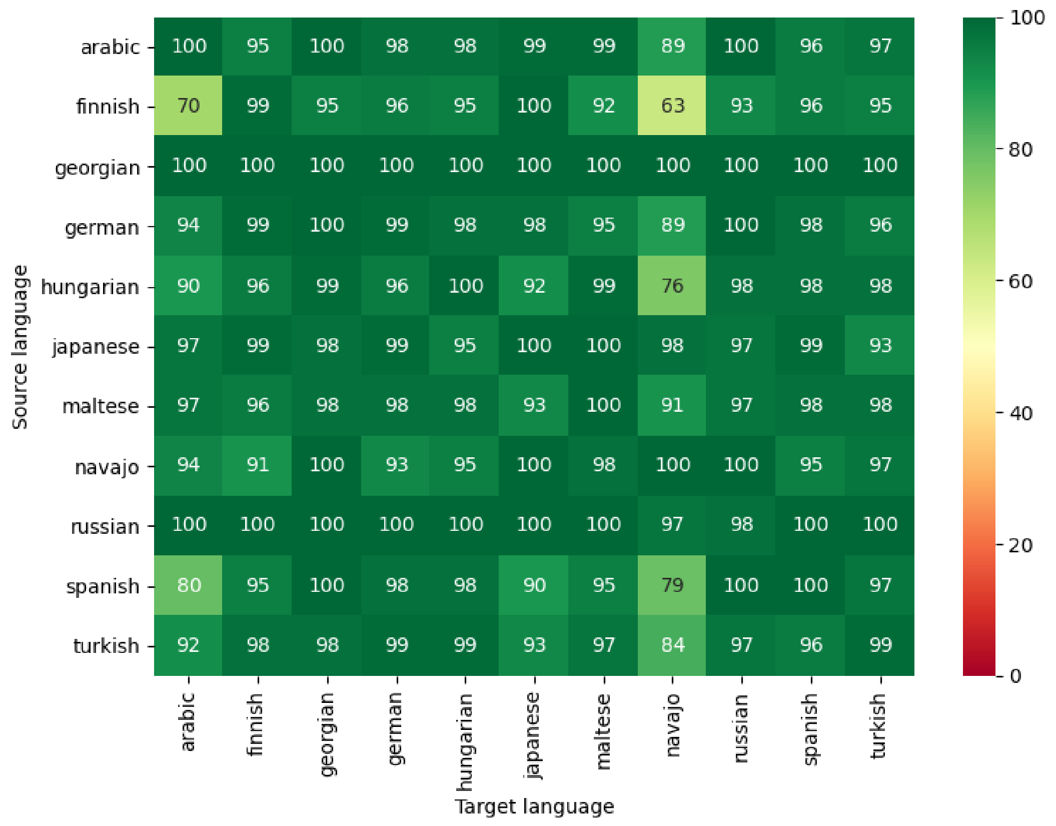
These two bilingual models offer opposite configurations: Hungarian-Turkish maximizes the transferability of the model, while Hungarian-Finnish considers closely related languages which might negatively impact transferability.

For both models, we consider the partial and full transfer settings defined in Subsection 8.1.1.

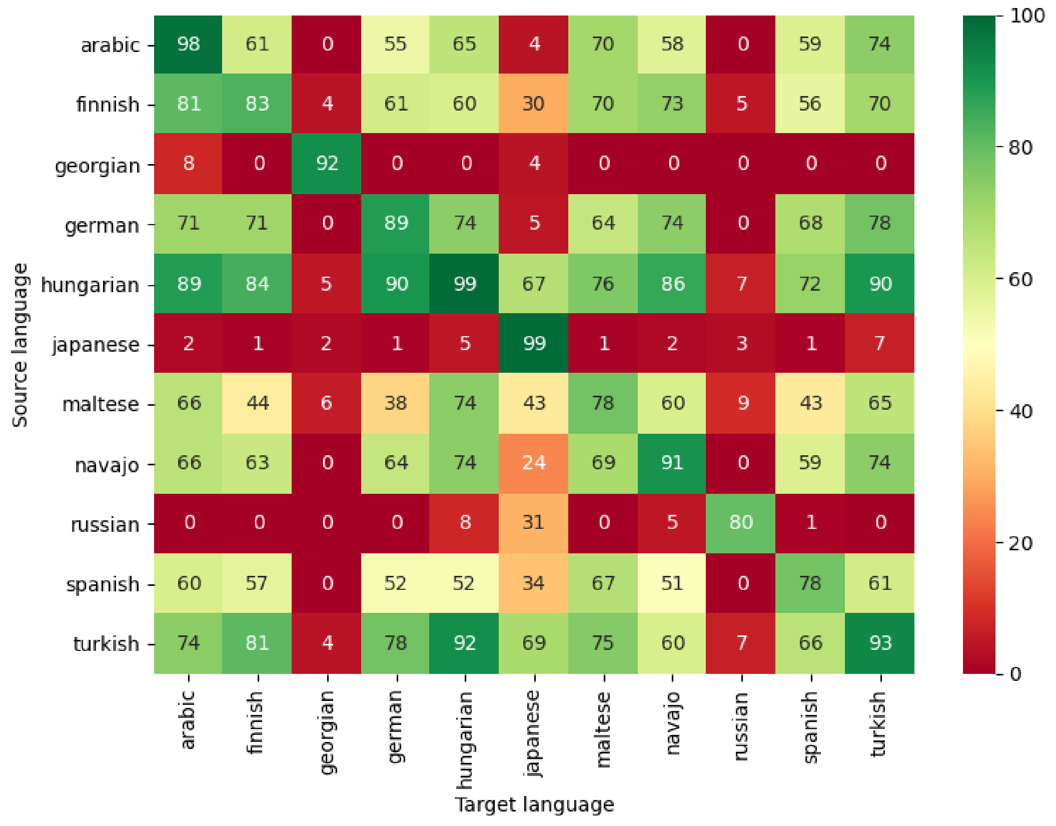
Variants of omnilingual models. We consider two sets of training languages for omnilingual models: one with and one without Japanese in the training data, due to how different Japanese is from the languages in Sig16. In particular, as languages in Sig16 are all romanized and Japanese uses the original characters, the characters used show no overlap which is a potential source of issues for the CNN-emb.

We consider two omnilingual CNN-emb variants with regards to the embedding model, as follows.

- We can use a single embedding model for all the languages. In that setting, the training set contains data from all training languages shuffled together, so that the model is not trained on only one language, then the second one, *etc.*. This measure is meant to avoid having the CNN-emb "forget" the morphology of the first languages seen, by the end of training.
- In the multi-embedding model setting, we train one embedding model per language but a single ANNc. Therefore, we use as many CNN-emb as languages in the training data. For this model, the training sets of data are concatenated one after another after being shuffled,

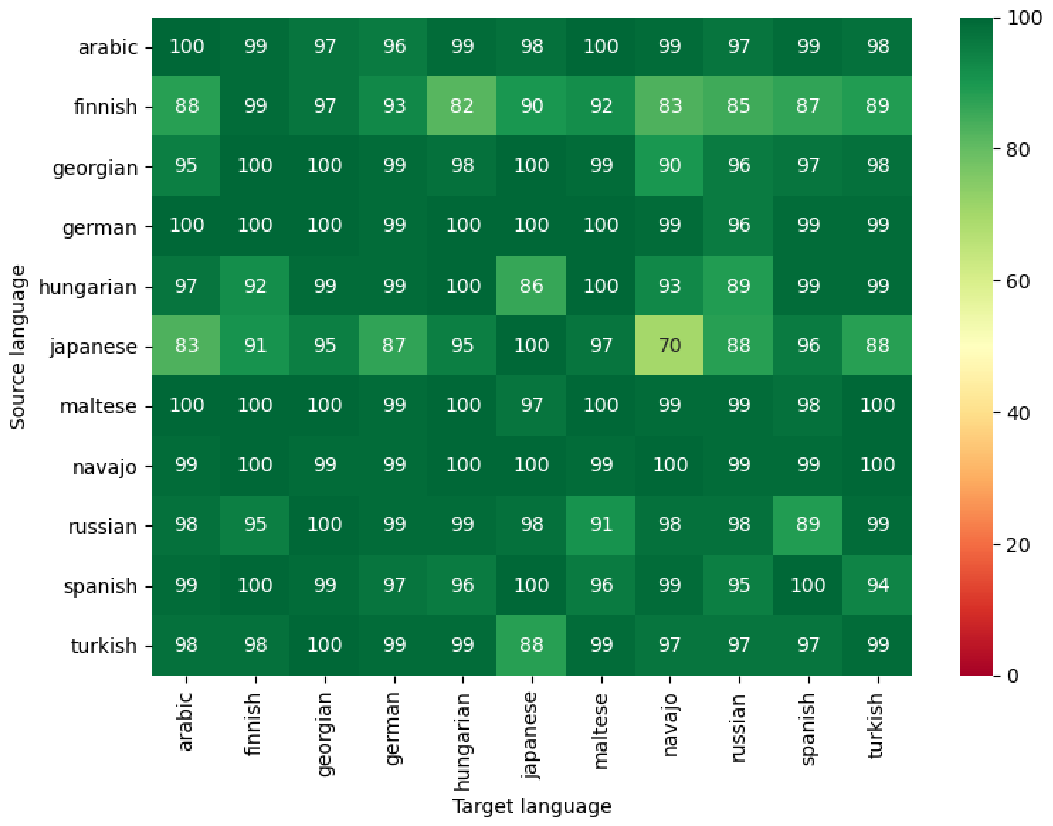


Source \ Target	arabic	finnish	georgian	german	hungarian	japanese	maltese	navajo	russian	spanish	turkish
arabic	100	95	100	98	98	99	99	89	100	96	97
finnish	70	99	95	96	95	100	92	63	93	96	95
georgian	100	100	100	100	100	100	100	100	100	100	100
german	94	99	100	99	98	98	95	89	100	98	96
hungarian	90	96	99	96	100	92	99	76	98	98	98
japanese	97	99	98	99	95	100	100	98	97	99	93
maltese	97	96	98	98	98	93	100	91	97	98	98
navajo	94	91	100	93	95	100	98	100	100	95	97
russian	100	100	100	100	100	100	100	97	98	100	100
spanish	80	95	100	98	98	90	95	79	100	100	97
turkish	92	98	98	99	99	93	97	84	97	96	99



Source \ Target	arabic	finnish	georgian	german	hungarian	japanese	maltese	navajo	russian	spanish	turkish
arabic	98	61	0	55	65	4	70	58	0	59	74
finnish	81	83	4	61	60	30	70	73	5	56	70
georgian	8	0	92	0	0	4	0	0	0	0	0
german	71	71	0	89	74	5	64	74	0	68	78
hungarian	89	84	5	90	99	67	76	86	7	72	90
japanese	2	1	2	1	5	99	1	2	3	1	7
maltese	66	44	6	38	74	43	78	60	9	43	65
navajo	66	63	0	64	74	24	69	91	0	59	74
russian	0	0	0	0	8	31	0	5	80	1	0
spanish	60	57	0	52	52	34	67	51	0	78	61
turkish	74	81	4	78	92	69	75	60	7	66	93

Figure 8.1: Appendix Figure 4(b) and 4(c) from [Als+21b]. Accuracy (in %) of fully transferred models, on Sig16 and JBATS. The results are split in valid (top) and invalid (bottom) permutations, respectively on the top and bottom respectively. The source language L_{training} is the vertical axis, and the target of the transfer L_{test} is the horizontal axis.



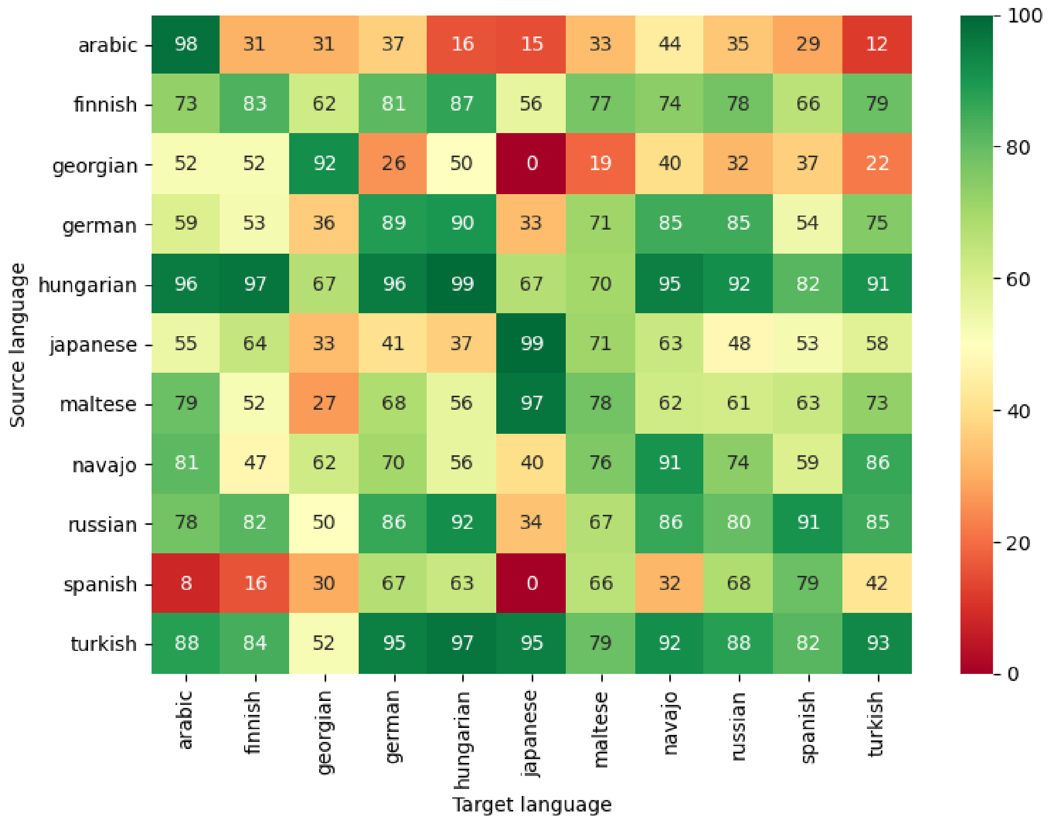


Figure 8.2: Appendix Figure 5(b) and 5(c) from [Als+21b]. Accuracy (in %) of partially transferred models, on Sig16 and JBATS. The results are split in valid (top) and invalid (bottom) permutations, respectively on the top and bottom respectively. The source language L_{training} is the vertical axis, and the target of the transfer L_{test} is the horizontal axis.

as the above precaution seems unnecessary. A special value is used during training to identify the language of the analogy.

When a single CNN-emb is used for all languages, we use $m = 512$ if Japanese is included and $m = 64$ otherwise.

Remark 8.1

In hindsight, we would recommend for the multi-embedding model setting to also use shuffling of the dataset across languages. Indeed, the CNN-emb models can not "forget" the morphology of the first languages seen anymore as they are not updated when processing data from other languages. By contrast, the problem transfers to ANNc, which will likely "forget" the configuration of the embedding space learned for the first languages seen, by the end of training.

Result for the bilingual models. The performance of the bilingual models is reported in Figures 8.3 and 8.4. Overall, performance is very high on valid APs (including the base forms) but lower on invalid permutations, with the recurring exception of Japanese, Navajo, and Arabic. This bias is similar to the one observed when performing transfer (see Subsection 8.1.1) and is consistent with the imbalance in favor of the valid APs of the 8/3 setting used for training.

For full transfer, performance is high when the bilingual model is transferred to one of the languages used to train the model. We also observe lower performance for invalid permutations in Russian and Georgian, but also for Japanese in particular with the Hungarian-Finnish model. To be more specific, the two bilingual CNN+ANNc models almost systematically classify the APs as valid, which results in 100% accuracy for the base and valid permutations and 0% to 2% for invalid permutations. German, Maltese and Spanish display the same kind of behavior, although to a smaller degree. Navajo and Arabic have an overall poorer full transfer performance than other languages, but show less imbalance between valid and invalid APs.

For partial transfer, performance when the bilingual model is transferred to one of the languages used to train the model is slightly lower than for full transfer. The Hungarian-Finnish model performs overall better than the Hungarian-Turkish model. In comparison, the latter had less consistent behavior in partial transfer, with in general noticeably lower performance for the base forms than the other valid permutations, in particular for Japanese.

Result for the omnilingual models. The accuracy results for the omnilingual models are reported in Figure 8.5 for the settings with a shared embedding model for all languages, and in Figure 8.6 for the model with separate embedding models. The performance of the four models is comparable across all setting and all languages, with two exceptions. First, for Maltese the performance is lower than those of other languages. Second, when Japanese is used, it appears to have higher overall performance than other languages. Other than this higher performance on Japanese, including or excluding the latter appears to have no effect on the performance of the model.

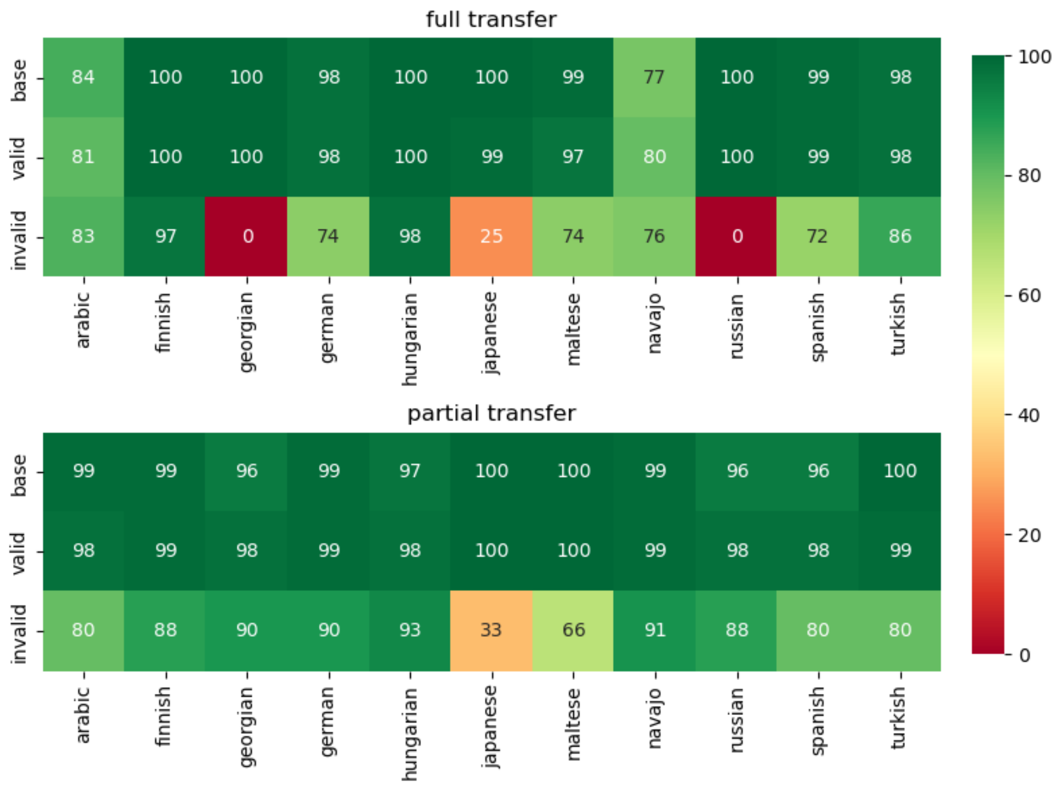


Figure 8.3: Appendix Figure 3(a) from [Als+21b]. Accuracy (in %) of full and partial transfer for the Hungarian-Finnish bilingual model, on Sig16 and JBATS.

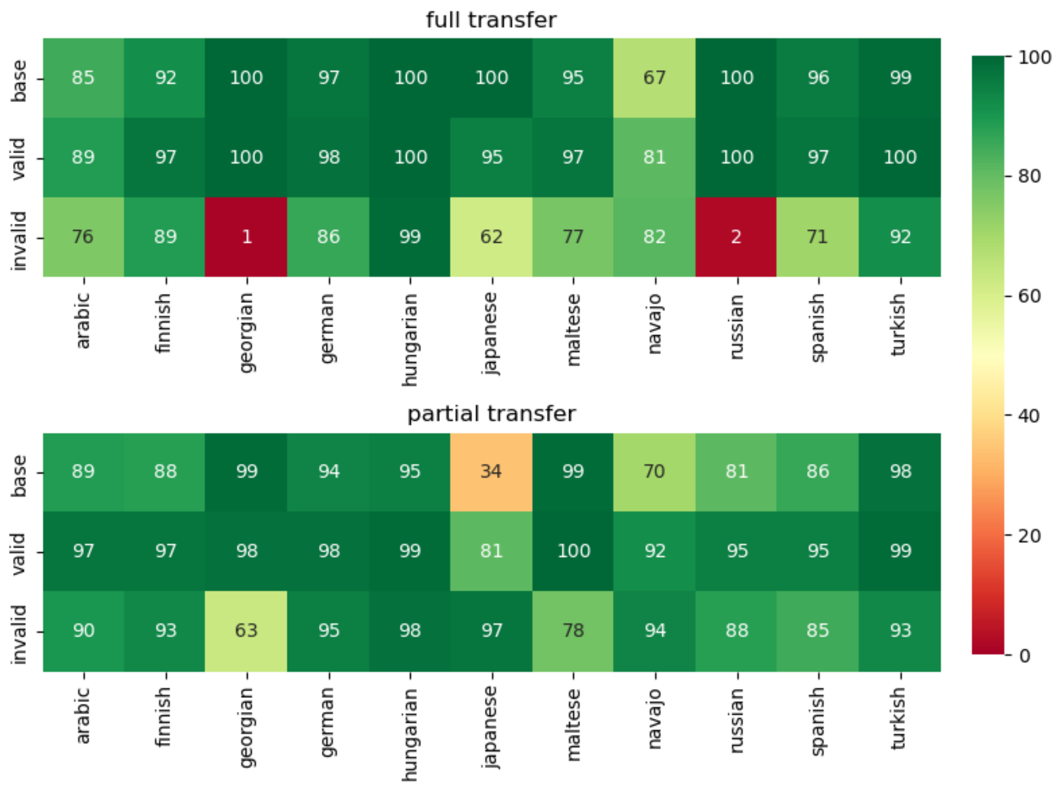


Figure 8.4: Appendix Figure 3(b) from [Als+21b]. Accuracy (in %) of full and partial transfer for the Hungarian-Turkish bilingual model, on Sig16 and JBATS.

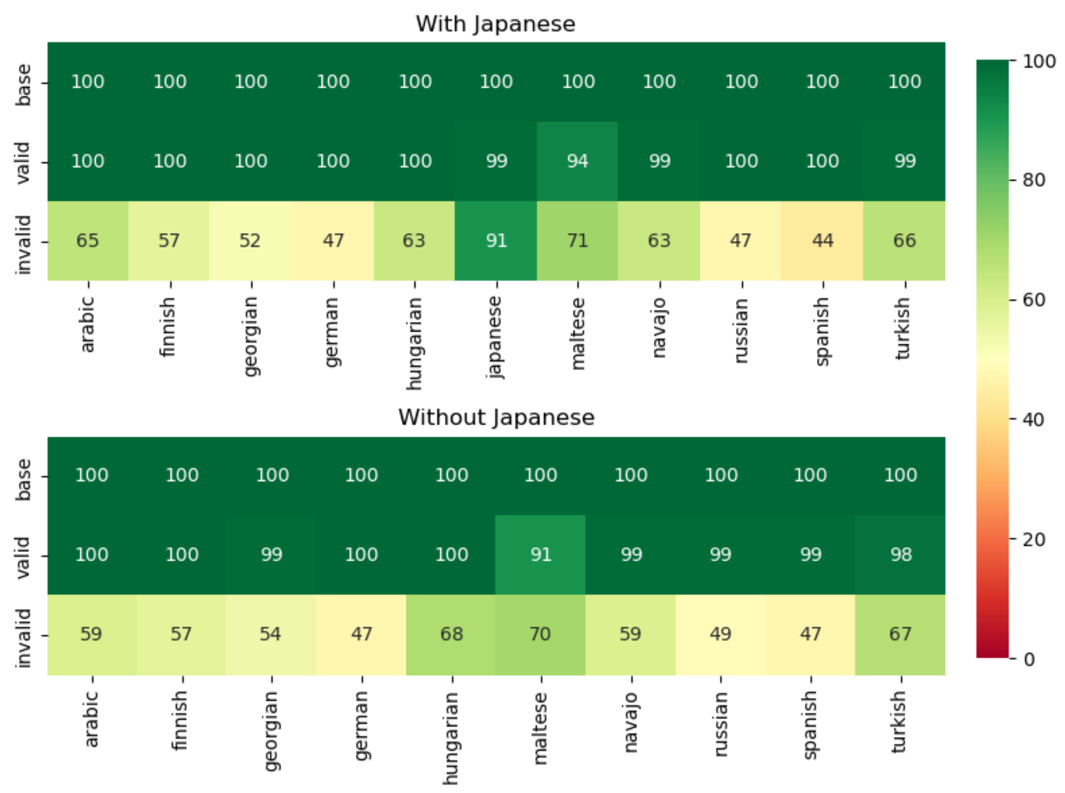


Figure 8.5: Appendix Figure 4(a) from [Als+21b]. Accuracy (in %) of full transfer for the omnilingual models with a single embedding model for all languages, on Sig16 and JBATS.

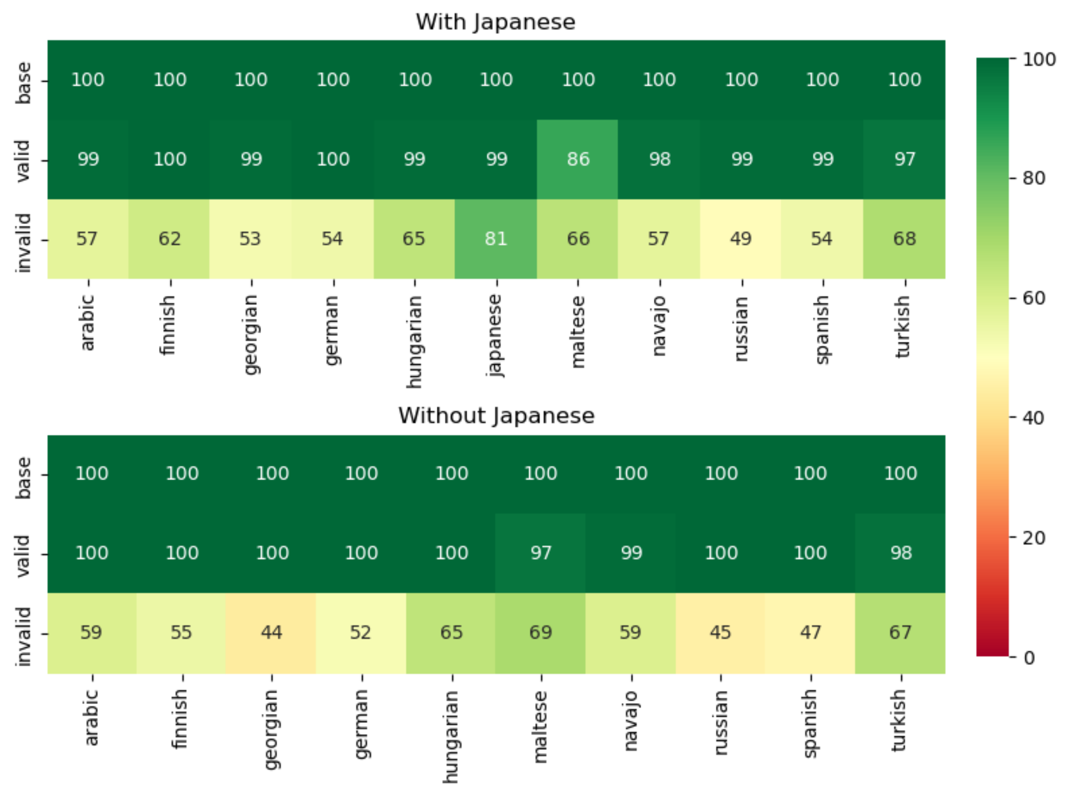


Figure 8.6: Appendix Figure 4(b) from [Als+21b]. Accuracy (in %) of full transfer for the omnilingual models with one embedding model per language, on Sig16 and JBATS.

8.2 Transfer performance and language families in Sig19

To go beyond the limitations of character coverage, in [MMC22] we leveraged the variety of languages available in Sig19 to reiterate the cross-lingual transfer experiments from [Als+21b, see also Subsection 8.1.1] at a larger scale. We were able to perform transfer between languages with similar alphabets which significantly reduced the impact of the alphabet gap.

Selection of the languages. For the experiment we only considered the high resource languages of Sig19, excluding Basque and Uzbek as they have less than the 50 000 base APs we use to train CNN+ANNc.

At the time of the experiment, we already identified the issue of the alphabet gap for cross-lingual transfer from previous cross-lingual transfer experiments (detailed in Subsection 8.1.1) and the transfer between Sig16 and Sig19 (see Subsection 7.3.4). In particular, we identified in the latter the correlation between the character coverage and the transfer performance. To limit the impact of the alphabet gap in this experiment, we extracted clusters of languages sharing a significant part of their alphabets, and transfer only within each cluster.

In Figure 8.7 we report for all 42 languages the character coverage of the test (target) language by the training (source) language. To perform hierarchical clustering, we use the character Jaccard (see Subsection 7.1.1) instead of coverage, as a symmetric distance matrix is required. We use the nearest point algorithm (or "single linkage") to get the clusters, and provide the associated dendrogram in Figure 8.8. Using a threshold of 0.4 on the Jaccard index of characters, we extract four clusters of at least two elements, colored in the dendrogram. With further analysis of the languages in each cluster, we named the clusters after the dominant alphabetic setting of the languages it contains. Based on what we observe in Figure 8.7 we found relevant to include Romanian in both the Roman and Cyrilic clusters, and obtain the following clusters:

1. **Roman** cluster: Albanian, Asturian, Czech, Danish, Dutch, English, Estonian, Finnish, French, German, Hungarian, Irish, Italian, Kurmanji, Latin, Latvian, Polish, Portuguese, Romanian, Slovak, Slovene, Sorani, Spanish, Swahili, Turkish, Welsh, and Zulu;
2. **Cyrillic** cluster: Adyghe, Bashkir, Belarusian, Bulgarian, Romanian, and Russian;
3. **Arabic** cluster: Arabic, Persian, and Urdu;
4. **Devanagari** cluster: Hindi and Sanskrit.

Limiting the experiments to transfers within clusters allows us to omit transfers likely to perform poorly due to the alphabet gap. It also reduces the number of transfers to attempt from $42 \times 41 = 1722$ to 740 transfers, excluding cases where the training and testing language is the same.

Training and test settings. Training is done in the balanced 8/8 setting (see Subsection 6.3.3), and balanced accuracy is used to measure performance. Due to the large number of experiments to perform, a single random seed is used for each model, by contrast with our experiments in Chapter 7. We also reduce the number of base APs used for testing from 50 000 to 5 000, and use the default 8/24 setting for the data augmentation of the test set.

Transfer performance beyond the alphabet gap. Once the languages with an overlap lower than 40% are eliminated, the Pearson correlation coefficient r between the performance and the character coverage drops compared to what was observed in Subsection 7.3.4. For the interpretation of the Pearson correlation coefficient we use the notions from [Ako18, Table 1]. As a reminder, we reported $r = 0.9639$ and $r = 0.7595$ for the transfer performance between the Sig16 and Sig19 versions of the same languages. We reported $r = 0.9739$ and $r = 0.8639$ when normalizing the transfer performance by the performance of the model trained on the test language. These are are strong to very strong correlations.

Due to their small size relative to cluster 1, we group clusters 2, 3, and 4 when computing correlations. For clusters 2, 3, and 4, we observe $r = 0.6565$, which can be seen as a moderate correlation. For cluster 1, the largest cluster, coverage and performance appear uncorrelated with $r = 0.0380$. Similar values are observed when normalizing the transfer performance of the model by the performance trained on the test language: $r = 0.6199$ for clusters 2, 3, and 4, and $r = -0.0579$ for cluster 1.

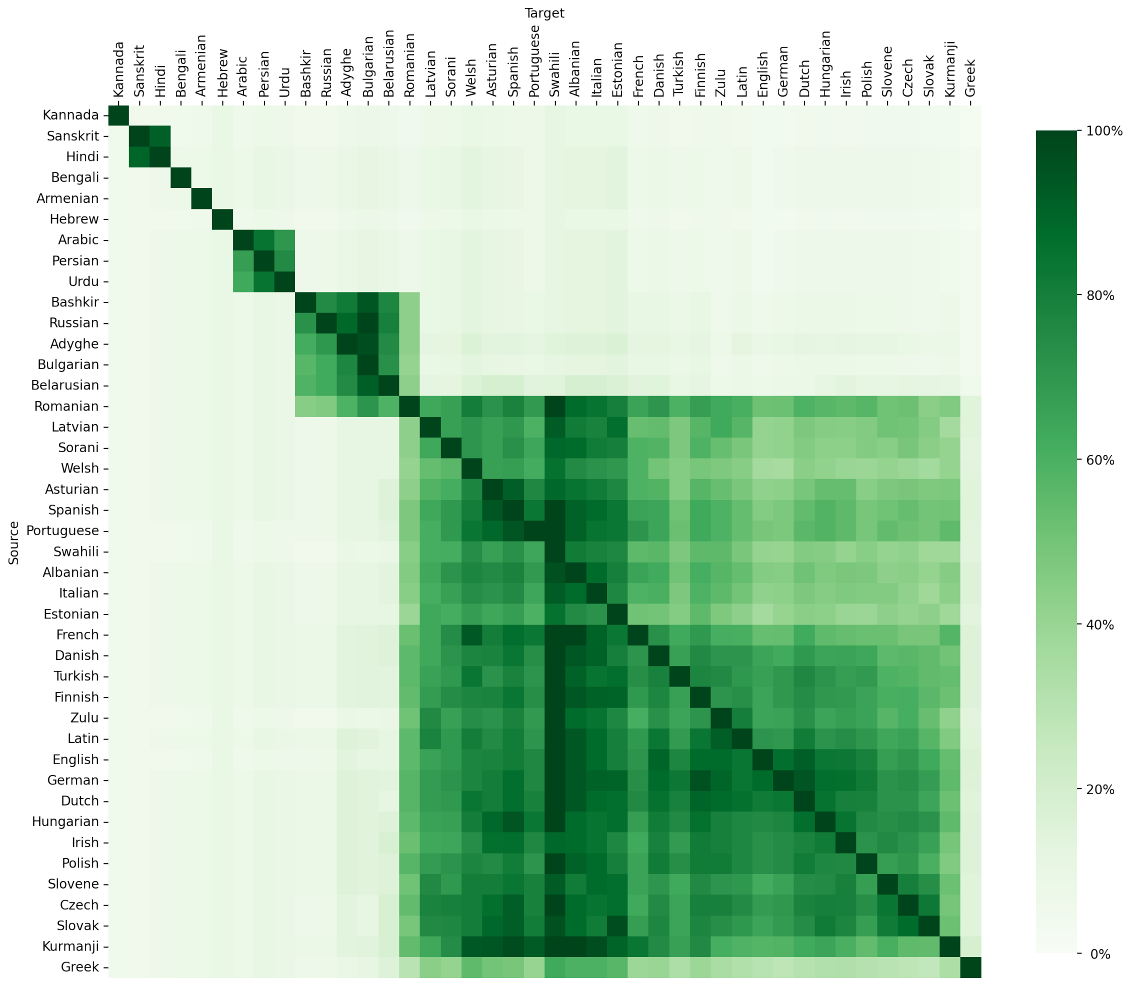


Figure 8.7: Appendix Figure 1 from [MMC22]. Coverage by the source language character vocabulary of the target language character vocabulary.

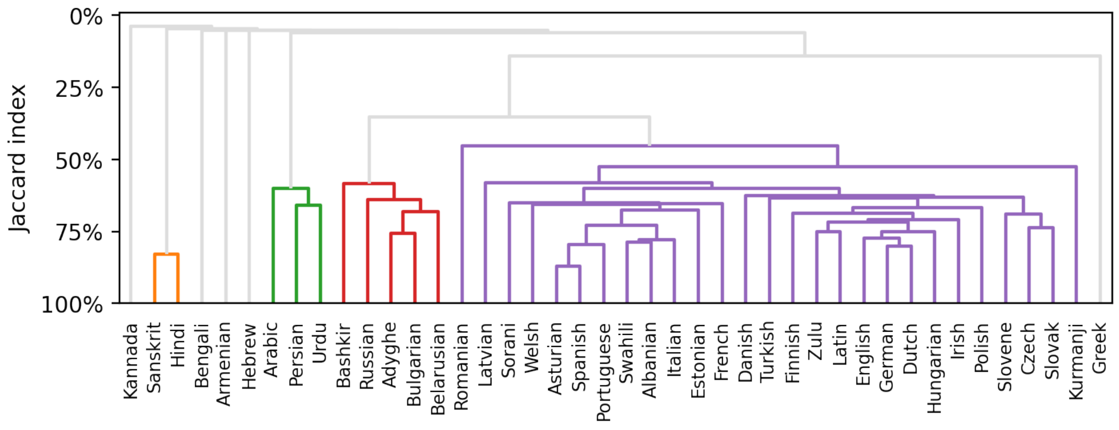


Figure 8.8: Appendix Figure 2 from [MMC22]. Dendrogram of the high resource languages in Sigmorphon2019 (except Basque and Uzbek), based on the Jaccard index between each pair of languages. With a threshold of 40% on the Jaccard and excluding singletons, four clusters (colored here) are found.

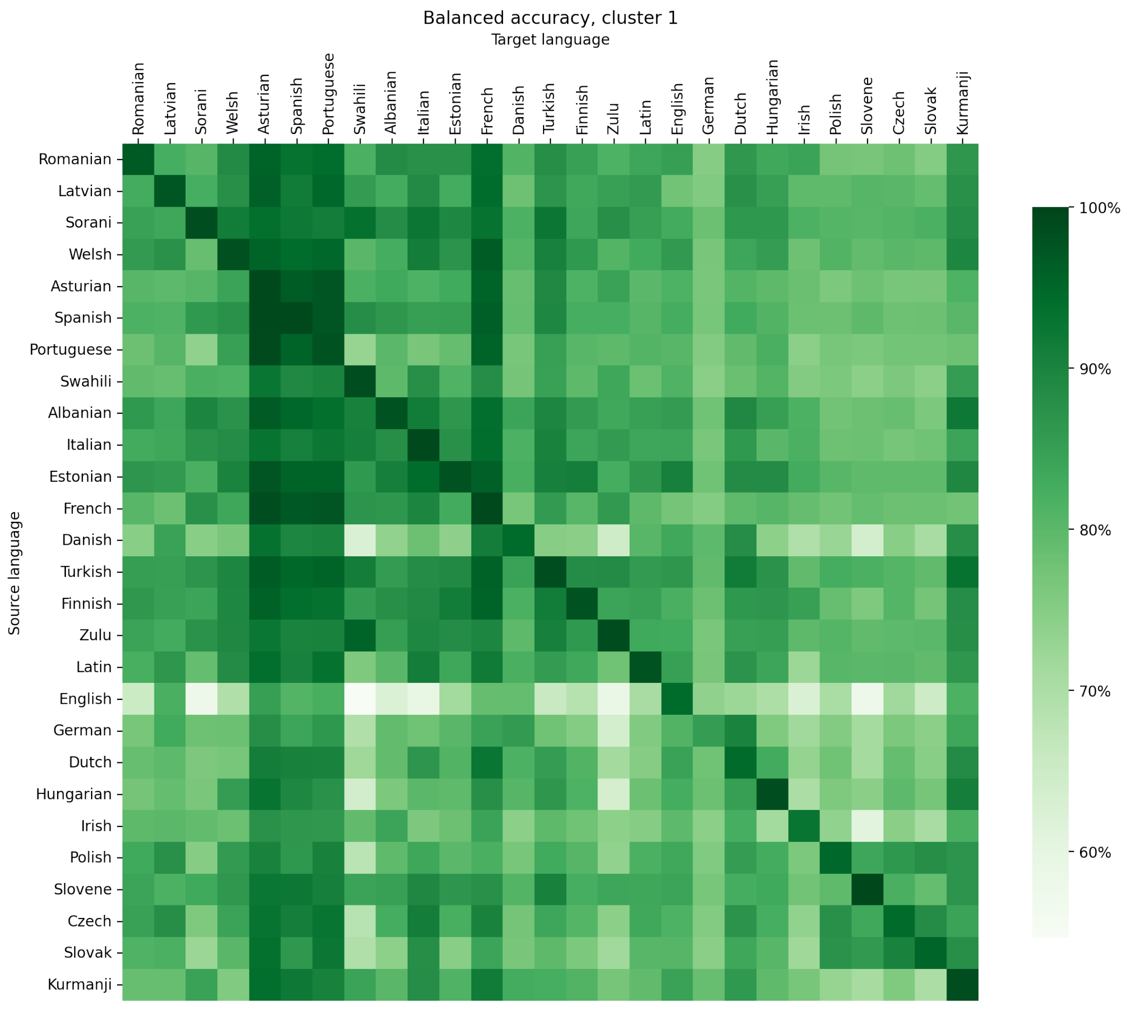


Figure 8.9: Figure 3 from [MMC22]. Transfer accuracy within cluster 1.

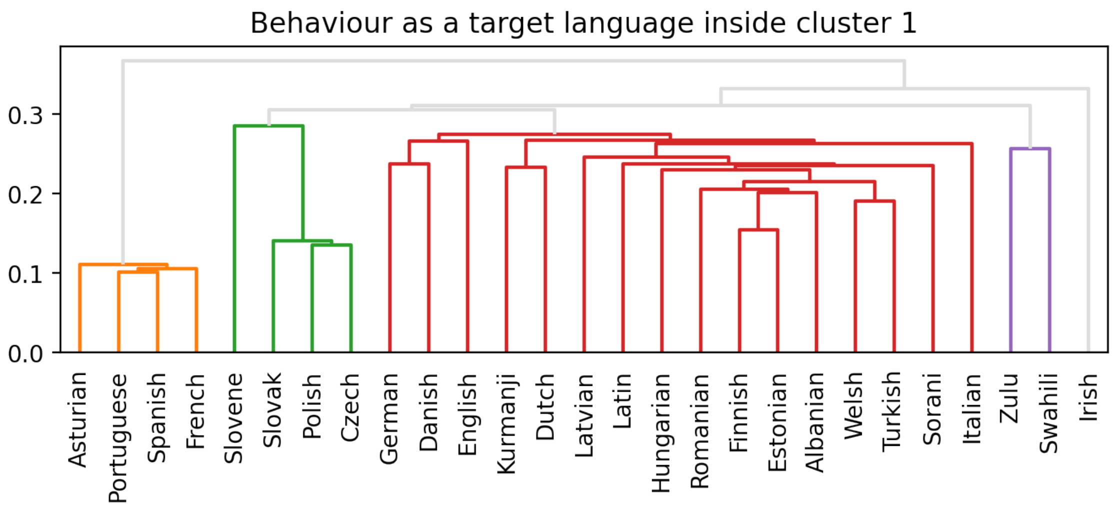


Figure 8.10: Figure 4 from [MMC22]. Dendrogram of the target languages, based on the transfer accuracy from all source languages as features for the target languages.

While these correlations might be influenced by the smaller amount of data used (only one seed, fewer testing analogies than usual), it is unlikely that such a bias is the cause of such a significant drop in correlation.

Deeper analysis of transfer performance in cluster 1. We report transfer performance for cluster 1 in Figure 8.9. The tendencies we observe in the transfer performance for cluster 1 in Figure 8.9 indicate that the performance is linked to the language being used as a source language or as a target. For instance, we observe a distinct horizontal bar for English and vertical bars for Asturian and German.

This behavior is likely due to either *(i)* the quality of the learned CNN+ANNc (how well it performs in general) or *(ii)* to the morphological similarities of some languages within Sig19. The former hypothesis is less likely, as only tendencies in the behavior as a training language (*i.e.*, horizontal bars) would be observed, while we mostly observe tendencies in the behavior as a test language (*i.e.*, vertical bars).

To confirm the influence of language similarities on performance, we explore hierarchical clustering within cluster 1 to study which key groups appear. When considering the behavior of the language as a test language, *i.e.*, using performance from different training languages as a features for the clustering, the clusters are more distinct than if we consider the performance as a training language. Therefore, we focus on clusters extracted from the former, which can be seen in the dendrogram in Figure 8.10.

We find that the small clusters, which are the most easily distinguishable by the clustering algorithm, correspond to closely related groups of languages. To identify relatedness of languages, we use the notion of language families, extracted from the "Language family" field of the infobox in the Wikipedia page of each language. To make the parallel more striking, we represent in Figure 8.11 the tree containing the language families of the languages in cluster 1. In this tree, we use the same color for the languages as the color, in Figure 8.10, of the cluster they belong to.

More precisely, we observe that the **orange** cluster contains Western Romance languages (Asturian, Portuguese, Spanish, and French), the **purple** cluster contains all the Bantu languages (Zulu and Swahili), and the **green** cluster contains Slavic languages: West Slavic languages (Slovak, Polish, and Czech) and slightly further the South Slavic language (Slovene). Irish is isolated. Finally, the **red** cluster contains all the remaining languages, even if distinct sub-clusters can be found in Figure 8.8: the Finnish and Estonian sub-cluster corresponds to Finnic languages and the Romanian and Italian sub-cluster contains the non-Western Romance languages. Other sub-clusters of the **red** cluster do not correspond to specific language families, like the Kurmanji and Dutch and the Welsh and Dutch sub-clusters.

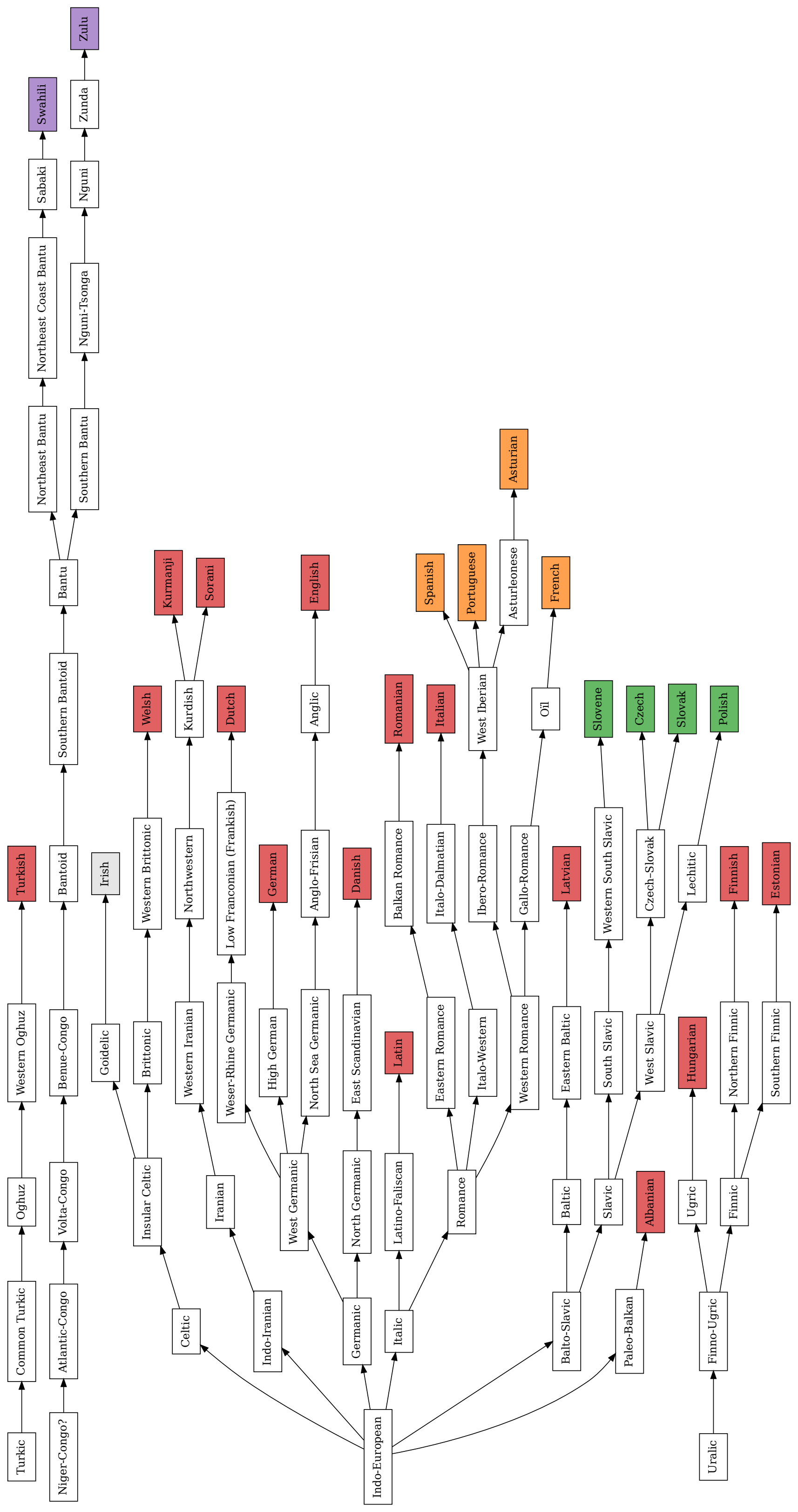


Figure 8.11: Appendix Figure 3 from [MMC22]. Trees of how the languages in cluster 1 relate based on their "Language family" according to Wikipedia. The Wikipedia page of each language contains an infobox (the area containing key information about the topic of the page), from which we extracted the "Language family" field. Languages are highlighted to match the clusters in the dendrogram of cluster 1.

8.3 Discussion and perspectives

We performed several experiments on using CNN+ANNc to model multiple languages at once (see Subsection 8.1.2) and to transfer CNN+ANNc between languages without fine-tuning (see Subsection 8.1.1 and Section 8.2), leveraging the multilingual nature of Siganalogies.

We made a first attempt to study how transfer performance relates to family in [Als+21b]. However, the first experiments on Sig16 and JBATS did not allow us to draw general conclusions beyond confirming the limitations caused by the alphabet gap. We also observed that the imbalance caused by the data augmentation without sampling is revealed through transfer to other languages, even when performance on the training language appears satisfactory.

8.3.1 Multilingual models

As with models trained on a single language, we found that the imbalance caused by the 8/3 data augmentation setting transfers to other languages when using multilingual models. The multilingual models we considered in Subsection 8.1.2 offer more stable performance across languages compared to models trained on a single language, with comparable amounts of training data. While the omnilingual models offer more stable performance than bilingual models, the bilingual models perform much better as long as we exclude two problematic languages from Sig16 as well as JBATS.

8.3.2 Cross-lingual transfer

Transferring the whole CNN+ANNc resulted in reasonable performance (often above 50% accuracy) on Sig16, however the transfer failed on some languages (see Subsection 8.1.1). Indeed, those languages used a different set of characters than the rest of languages, resulting in a significant amount of unrecognized characters and causing an alphabet gap (that we also describe in Subsection 5.5.1). We performed further transfer experiments on Sig19 (see Section 8.2), in which we were able to circumvent the alphabet gap issue and achieved higher success.

To be more specific, with the cross-lingual transfer experiments performed on Sig19, we confirmed that under the right circumstances, *i.e.*, when alphabet gap is not limiting anymore, the morphological similarities between languages are reflected in the behavior of CNN+ANNc during transfer. By extension, the morphological transformations modeled by CNN+ANNc through APs appear transferrable across languages.

Nevertheless, the transfer performance is most likely influenced by the extent of the morphology of a language present in the data, as Sig19 does not represent the full morphology of each language. This might explain why transfer in some clusters of languages does not perform as well as in the largest cluster we studied.

The experiments on Sig19 revealed an interesting correlation between the transfer performance and the proximity of the source languages in the Wikipedia language families. In particular, we found that in many cases, it is possible to use the proximity in the Wikipedia language families to predict the performance of transferred models, as models trained on closely related languages will perform similarly when transferred.

8.3.3 Future work

The results we obtain on Sig19 offer new perspectives for the development of multi-lingual models, for instance by training separate models for each cluster of languages we identified in Sig19 based on the usage of similar alphabets. We think that using only a subset of relevant languages for training could improve results and offer a global morphological analogy model for a relatively cheap cost in terms of data. Extensions to cross-lingual APs would be an interesting application of our results, as this would open the use of ANN framework to languages with few written ressources available.

In our experiments, we considered only APs with all the elements from the same language. There exist a type of cross-lingual analogy we did not explore, where two of the elements come from one language and the other two from another language. It would be interesting to use the low ressource languages from Sig19 as an experimental setting for such analogies, as we could attempt to use high ressource languages as a base for the analogical transfer of morphological transformations.

Finally, our multilingual experiments were performed using only CNN+ANNc. A natural direction for future work is to explore the performance of ANNr in a similar multilingual setting.

Chapter 9

Conclusion of Part II and dissemination of the ANN framework

We conclude this Part II by summarizing our contributions, and explain the various means put in place to disseminate our work to the general public and the scientific community.

Our Siganalogies dataset is discussed in more details in Section 5.5. The ANN framework we propose for analogy detection and analogy solving uses ANNc and ANNr, two DL models inspired from the properties of APs. These models are discussed in more details Sections 6.5, 7.7 and 8.3.

Contributions. The ANN framework achieves very high analogy detection and analogy solving performance, in a variety of settings. Both the CNN+ANNc for analogy detection and the AE+ANNr for analogy solving achieve consistent SotA performance on all the languages used in our experiments, and outperform the symbolic baselines Alea [LYZ09] and Kolmo [Mur+20] by as much as 80% of accuracy in some cases. This performance is, to some extent, transferable between languages.

The higher performance results from, among other things, *(i)* a representation of words learned from the data for the purpose of analogy manipulation, *(ii)* the ability of the model to integrate the dependencies between the embedding dimensions, and *(iii)* the flexibility to go beyond arbitrary arithmetic formulas of APs such as 3CosAdd, 3CosMul, or parallelogram rule. However, there is a known trade-off between the performance of DL models and their interpretability. In particular, it is usually difficult to understand why a NN obtains a particular output. This also applies to our framework, but further work might provide theoretical guarantees or empirical methods to tackle this limitation.

The framework leverages the properties and intuitions of APs in the design of the models, but also to augment data in a way that benefits model performance and sensitivity to initial conditions. The model also learns to be invariant with regards to permutations following the axioms of APs. In settings where APs appear ill suited due to some of the postulates, the data augmentation can be adapted, as we show in Section 7.6 for Exchange of the Means. We propose a generalization of this process in Appendix D, which will be supported by experiments in further work.

Overall, our framework shows that it is possible to obtain high performance when manipulating APs beyond arithmetic models on embeddings or manually-designed approaches. The approach can be applied to other types of data following our guidelines, as can be seen in [Als+22b; JCM23; LPR19; LPR21; Zer+22] and Chapters 10 and 11.

Dissemination. To make our work on the ANN framework for morphology available in the literature, we published in several international workshops [Als+21b; Cha+22; MMC22] and conferences [Als+21a; Mar+22a], and in a journal [MC24]. We also made a significant effort to make the research reproductible.

First, as mentioned in Section 5.4, all the data we use is made available on Dorel[1] and the code to produce and manipulate it on GitHub[2]. The dataset was the subject of a notice[3] in the

[1]https://dorel.univ-lorraine.fr/dataset.xhtml?persistentId=doi:10.12763/MLCFIE

[2]https://github.com/EMarquer/siganalogies

[3]https://recherche.data.gouv.fr/en/dataset/siganalogies-millions-of-morphological-analogies-in-more-than-80-languages

recherche.data.gouv.fr website, to increase its visibility.

Several GitHub repositories have been put in place along our experiments, the latest being the one for reproducing [MC24]. All the models trained for [MC24] are made available in Dorel[4].

Finally, the models from [MC24] are publicly made available for execution for the general public through the *ANNa* endpoint[5] hosted at Loria, that I initially developed as a proof of concept and was made fully operational through the combined efforts of several collaborators at Loria[6]. The website contains an online demo of the models, where it is possible to play around with APs and analogical equations in the languages mentioned in [MC24], and try out the prediction of CNN+ANNc for classification, CNN+ANNr for retrieval, and AE+ANNr for generation. This endpoint is bound to be integrated in the AT2TA online platform, in the frame of the ANR project "Analogies: from Theory to Tools and Applications" (AT2TA, grant number ANR-22-CE23-0023)[7]. The platform will integrate other tools and ressources for analogical reasoning developed within the AT2TA project.

[4]https://dorel.univ-lorraine.fr/dataset.xhtml?persistentId=doi:10.12763/I5ED78

[5]https://anna.loria.fr/morpho

[6]Many thanks to Miguel Couceiro, Christophe Cerisara, and Nousradine Cherif Hassan for their help and maintenance on the *ANNa* website, see https://anna.loria.fr/about.

[7]https://at2ta.loria.fr/software/

Part III
Beyond morphological analogies

This part describes our work done on some adaptations of the ANN framework presented in Part II to other domains. It also cover an application of APs to CBR. Our publications covered in this section are [Mar+23; Zer+22].

Chapter 10

Extension of the ANN framework to Target Sense Verification

Chapter contents

This chapter summarizes the contributions made in [Zer+22], and follows the same content. Further experiments and extended descriptions are available in the Ph.D. thesis of Zervakis [Zer23].

The Target Sense Verification (TSV) task is a type of Word Sense Desambiguation (WSD), that consists in determining whether the sense of word in a given context (intended sense) matches one of the possible sense of the word (target sense). Three elements are provided for the task: on the one hand, the word in its context; and on the other hand there is a definition and hypernyms (*i.e.*, words with a more general meaning than the target word, but covering the sense of the latter) for one possible sense of the word. In [Zer+22], we tackle TSV by combining BERT and ANNc and reformulating TSV as a analogy detection task.

Our experiments demonstrate the significant impact on the final performance of the position of the definition and hypernyms in the input of BERT, as well as how emphasis is made on the hypernyms. Moreover, we achieve competitive results on the Words-in-Context-TSV (WiC-TSV) evaluation benchmark [Bre+21]. Finally, we experiment with and without analogical data augmentation from Section 6.3, and observe that analogical data augmentation yields comparable performance and alleviates the dependence on the input encoding of BERT.

10.1 The task of Target Sense Verification

TSV data in WiC-TSV. For the TSV task we use the WiC-TSV dataset [Bre+21]. It contains pairs of two senses S_I, S_T, respectively the intended and target senses, with a label stating whether S_I matches S_T, the two senses correspond respectively to the target word in its context for S_I, and the definitions and hypernyms for S_T. Examples of such data are reported in Table 10.1.

The TSV task is divided into three sub-problems, depending on what is used to represent sense S_T:

- only the definition (sub-task 1);
- only the hypernyms (sub-task 2);
- both (sub-task 3).

Intended sense	**Target sense**		
Context	**Definition**	**Hypernyms**	**Label**
A $\underbrace{\text{[marriage]}}_{\text{target}}$ of ideas.	A close and intimate union.	union, unification	True
A $\underbrace{\text{[fight]}}_{\text{target}}$ broke out at the hockey game.	The act of fighting; any contest or struggle.	conflict, struggle, battle	True
My neighbor was the lead $\underbrace{\text{[role]}}_{\text{target}}$ in last year's village play.	The actions and activities assigned to or required or expected of a person or group.	duty	False
They went bankrupt during the economic $\underbrace{\text{[crisis]}}_{\text{target}}$.	A crucial stage or turning point in the course of something.	juncture, occasion	False

Table 10.1: Examples taken from the development set of WiC-TSV.

	Instances	Total	Positive example rate
Train	WNT/WKT	2137	0.56
Dev	WNT/WKT	389	0.51
Test	WNT/WKT	717	0.54
	MSH	205	0.52
	CTL	216	0.43
	CPS	168	0.46

Table 10.2: Statistics of the WiC-TSV dataset.

In our experiments, we only consider sub-task 3.

WiC-TSV contains general-domain instances, extracted from WordNet (WNT) and Wikitionary (WKT). It also contains domain-specific instances:

- general-domain instances, extracted from WordNet (WNT/WKT);
- domain-specific instances:
  - Cocktails (CLT), extracted from the "All about cock-tails"[1] thesaurus;
  - Medical Subjects (MSH), extracted from the MeSH[2] thesaurus;
  - Computer Science (CPS), manually constructed from Wikipedia definitions and a consensus of 2 experts.

The training set and development set contain only general-domain sentences, while the test set contains sentences from all 3 sources, with the exact amounts reported in Table 10.2.

Baseline approaches to TSV. The authors of the WiC-TSV dataset proposed, in [Bre+21], a model based on BERT to solve the task. For sub-task 3, the concatenation of the context, definition, and hypernym is fed to BERT. Then, a binary classification layer (*i.e.*, a perceptron with a single output and sigmoid activation) is used to predict whether the senses match or not. The input to this classifier is the concatenation of:

- the `[CLS]` token embedding;
- the average of the embeddings corresponding to the target word;
- the average of the embeddings of the definition.

[1] http://vocabulary.semantic-web.at/cocktails
[2] https://www.nlm.nih.gov/mesh/meshhome.html

Breit, Revenko, et al. tested both the base and large versions of BERT, referred to as BERT-B and BERT-L, respectively. They also propose a FastText baseline using the concatenation of the average of the embeddings of the context, and the average of the embeddings of the definition. They also include unsupervised baselines based on BERT and DistilBERT [San+19], labeled U-BERT and U-dBERT respectively.

A similar approach to BERT-B and BERT-L was proposed by Moreno, Pontes, and Dias [MPD21], where two distinct BERT models are fine-tuned respectively on the hypernyms and the definition. In each case, the representation of the target sense is concatenated to the context, and the embedding of the `[CLS]` token is used for classification. At inference time, the output of both models are aggregated. In their experiments, the authors place special tokens `$` around the target word in the context, and separate the hypernyms with such characters too.

An extensive study of BERT for TSV is presented in [VSD21], including data augmentation, freezing the model parameters during fine-tuning, applying different pooling strategies to obtain the classifier input, and masking the target word in the context.

A more generic approach called MIRROWIC was proposed by Liu, Liu, et al. [Liu+21]. It was tested on multiple tasks including TSV. Their method uses contrastive learning to bring the representation of words in similar contexts closer and words in dissimilar context further apart. To evaluate their pre-trained model on TSV, they constructed manual templates involving the target word, the definition and/or the hypernyms (depending on the sub-task). Then the authors classify each instance based on the cosine similarity of the embedding of the target word in its original context and in the template.

10.2 Analogy detection for TSV

Analogy detection can be used to check that the sense S_I of the target word matches the target sense S_T of the definition and hypernyms.

For instance, it is possible to check that the relation $R(\text{target word}, \text{hypernyms})$ is equivalent to the relation $R'(\text{definition}, \text{hypernyms})$, in other words checking that the AP

$$\text{target word} : \text{hypernyms} :: \text{definition} : \text{hypernyms}$$

holds. Indeed, as the definition and the hypernyms always correspond to the same sense S_T, if R, R' are equivalent, then the sense S_I of the target word would also be the same as S_T. We provide an example of this formulation in Example 10.1.

Example 10.1: Translating TSV into analogy detection

Using the first instance in Table 10.1 as an example we have:

- context = $\text{“A } \underbrace{[\text{marriage}]}_{\text{target}} \text{ of ideas.”}$;
- target word = “marriage”;
- definition = “A close and intimate union.”;
- hypernyms = {“union”, “unification”}.

The question of whether “*target word*” : “*hypernyms*” :: “*definition*” : “*hypernyms*” holds can then be reformulated as:

> “Is the relation from “marriage” (in the sentence “A marriage of ideas.”) to “union” and “unification”, the same as the relation from “A close and intimate union.” to “union” and “unification”?”

The answer to that would be yes, as “a close and intimate union” is a specific kind of “union” or “unification”, and the same can be said for “marriage” in the sentence “A marriage of ideas.”

Let us now we consider the definition and hypernyms:

- definition = “The act of marrying; the nuptial ceremony.”;

- hypernyms = {“ritual”,“rite”}.

In that case, the AP does not hold anymore, as “The act of marrying; the nuptial ceremony.” is a special kind of “ritual” or “rite”, but “marriage” in the sentence “A marriage of ideas.” is not.

Potential analogical formulations of TSV. The above AP for TSV is not the only possible formulation. Tackling TSV based on analogical reasoning requires us to select the appropriate $A, B, C, D \in \mathcal{A}$ such that the analogy detection task on APs $A : B :: C : D$ yields good classification performance. In particular, for our experiments we consider $\mathcal{A} = \{cls, tgt, ctx, def, hyps, descr\}$ as possible elements of the analogy:

- cls: the embedding of the `[CLS]` token;
- tgt: the embedding of the target word;
- ctx: the average of the embeddings of all words in the context;
- def: the average of the embeddings of all words in the definition;
- $hyps$: the average of the embeddings of all hypernyms;
- $descr$: the average of the embeddings of all words in the definition and all hypernyms.

The selection of $tgt, ctx, def, hyps$ are the elements provided for the TSV task, and carry the senses to compare. Additionally, cls the embedding of `[CLS]` can generally be seen as a representation of the whole input, and is often used for classification tasks on the whole input of BERT [Bre+21; Dev+19; MPD21; VSD21]. Therefore, cls may capture key information both from context, and definition/hypernyms. Finally, inspired by [Bre+21], we also test for $descr$, which essentially treats definition and hypernyms as a whole rather than separate units. Note that the choice and arrangement of these candidates in the analogical formulation is independent from the input formatting described in a later paragraph.

The amount of APs $A : B :: C : D$ with $A, B, C, D \in \mathcal{A}$ is $|S|^4 = 1,296$. However, not all APs in this set are of interest to is, for instance $hyps : hyps :: hyps : hyps$ is unhelpful for the TSV. To ensure the APs are meaningful for TSV, we distinguish two subset of $\mathcal{A}$: $\mathcal{A}_1 = \{cls, tgt, ctx\}$ contain information coming from the context, while $\mathcal{A}_2 = \{cls, def, hyps, descr\}$ involves embeddings which reflect the information found in the definition and hypernyms. Note that since it represent the whole instance, cls belongs to both subsets. We use $\mathcal{A}_1, \mathcal{A}_2$ to define a set of rules to filter interesting analogical formulations, reducing them to 768:

- $(A \neq B) \wedge (C \neq D)$ excludes APs where embeddings on either side are identical;
- $\neg\Big[\big[(A, B, C, D \in \mathcal{A}_1) \wedge (A \neq cls)\big] \vee \big[(A, B, C, D \in \mathcal{A}_2) \wedge (A \neq cls)\big]\Big]$ excludes APs instantiated exclusively from $\mathcal{A}_1$ or $\mathcal{A}_2$, with the exception of cls that covers the whole input, and therefore ensures that information from both the context/target and the definition/hypernyms is present.

Analogical data augmentation. In our experiments we train models with and without the analogical data augmentation from Section 6.3. We indicate models with analogical data augmentation during training with a subscript pi standing for permutation invariance training.

Contrary to what is described in Section 6.3, both valid and invalid examples are already available for the analogy detection task. Therefore, we apply the 8 equivalent permutations on the valid and invalid examples separately, and we do not use the invalid permutations mentioned in Section 6.3. Equivalent permutations are included in the same training minibatch.

BERT input format and hypernym marking. In all our experiments, the target word is surrounded by focus characters in the context, following [MPD21]. The order and format in which the context, definition, and hypernyms are fed into BERT has a direct impact on the embeddings produced. For instance, in preliminary experiments using the `[CLS] context [SEP] definition ; hypernym, hypernym, ... [SEP]` input encoding format, analogical formulations including $hyps$ performed particularly poorly. This issue is likely due to the fact that BERT is pre-trained on sentences and a list of hypernyms is not a sentence. To mitigate this issue, we format the hypernyms using special tokens, and include their embeddings in the computation of *hyps*. Indeed, such special tokens are often used in the input of BERT to help models identify and store information about special segments of the input [MPD21]. We consider two types of special tokens to help BERT identify hypernyms as non-sentence inputs, both used with success in the literature [MPD21; Zha+19]: a focus character (fc) `$` or opening and closing entity markers (em) `[H]` and `[/H]`. Given the sensitivity of BERT to the order of elements in the input, we consider a variant (swap) of the input format exchanging the position of the definition and the hypernyms.

We end up with six variants of the input formatting, with examples given in Example 10.2:

- **default**: `[CLS] context [SEP] definition ; hypernyms [SEP]`;
- **default+fc**: `[CLS] context [SEP] definition ; $ hypernyms $ [SEP]`;
- **default+em**: `[CLS] context [SEP] definition ; [H] hypernyms [/H] [SEP]`;
- **swap**: `[CLS] context [SEP] hypernyms ; definition [SEP]`;
- **swap+fc**: `[CLS] context [SEP] $ hypernyms $ ; definition [SEP]`;
- **swap+em**: `[CLS] context [SEP] [H] hypernyms [/H] ; definition [SEP]`.

Example 10.2: Input formatting variants.

Using the first instance in Table 10.1 as an example we have:

- **default**: `[CLS] A $ marriage $ of ideas.`
  `[SEP] a close and intimate union ; union, unification [SEP]`;
- **default+fc**: `[CLS] A $ marriage $ of ideas.`
  `[SEP] a close and intimate union ; $ union, unification $ [SEP]`;
- **default+em**: `[CLS] A $ marriage $ of ideas.`
  `[SEP] a close and intimate union ; [H] union, unification [/H] [SEP]`;
- **swap**: `[CLS] A $ marriage $ of ideas.`
  `[SEP] union, unification ; a close and intimate union [SEP]`;
- **swap+fc**: `[CLS] A $ marriage $ of ideas.`
  `[SEP] $ union, unification $ ; a close and intimate union [SEP]`;
- **swap+em**: `[CLS] A $ marriage $ of ideas.`
  `[SEP] [H] union, unification [/H] ; a close and intimate union [SEP]`.

10.3 Experiments

10.3.1 Impact of the input encoding and analogical formulation

For each choice of input encoding and relation, we train our system 4 times using different random seeds, without analogical data augmentation, resulting in $6 \times 768 \times 4 = 18,432$ runs. Each run takes approximately 35 minutes on a Nvidia GTX 1080 Ti 11GB. Figure 10.1 shows the mean accuracy achieved across all 4 runs, for each input formats, and for each combination of elements for the analogical formulation sorted in ascending order of performance.

Overall, for all input formats there exist some A, B, C, D combinations that result in good performances. Using entity markers outperforms using focus characters on the development set, as can be seen in Figure 10.1. This might be due to using the same focus characters for the target word and for the hypernyms, which is less expressive than when using dedicated entity markers.

However, some formats appear more sensitive than others to the selection of the elements in the AP. In particular, not using focus characters (fc) or entity markers (em) significantly lowers performance, with the 400 worst formulations performing particularly poorly. These formulations contain *hyps* for at least one element, and removing them brings the distribution of results much

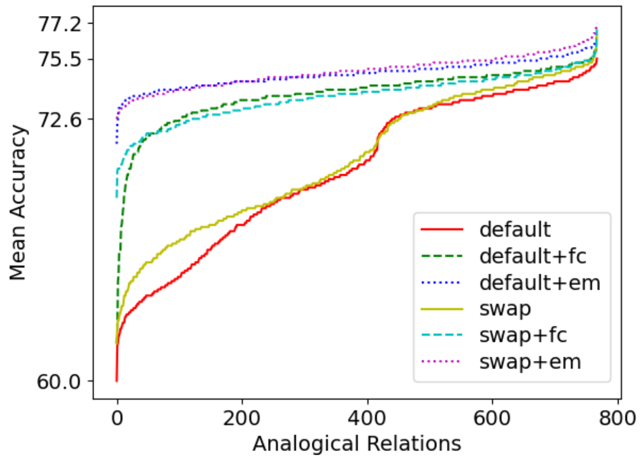


Figure 10.1: Mean accuracy achieved on the development set. Each curve represents a distinct input format, and the horizontal axis represents the 768 possible analogical formulations, sorted in order of increasing accuracy.

closer to the other two settings, which is consistent with our preliminary experiments. We also observed in additional experiments that if focus characters or entity markers are added in the formatting of hypernyms, but without including the embedding of said tokens in *hyps*, the performance is close to not using the focus characters or entity markers. This indicates that important information for TSV is stored in the embeddings of the special tokens.

By contrast, swapping or not the position of the definition and the hypernyms in the input does not have as much of an impact on performance.

For each input format, the best performing analogical formulation contains *cls*, which suggest that the `[CLS]` token is particularly important for the performance on TSV.

10.3.2 Comparison with other approaches to TSV

Based on the results of the previous experiment, we train AB4TSV models with and without analogical data augmentation, for 10 random initializations. We report the results on the development set in Table 10.3. We also reproduce the HyperBertCLS and HyperBert3 baselines from [Bre+21], using the 6 input formats we introduce. On the development set, AB4TSV outperforms all the other models. AB4TSV$_{pi}$ performs worse than AB4TSV within a margin of 1%, for both accuracy and F1 score. In terms of accuracy, putting the hypernyms before the definition (**swap**) and using focus characters or entity markers (**fc**/**em**) maximizes performance. In terms of F1 however, the best performing setting for AB4TSV and AB4TSV$_{pi}$ (**default+em**) puts the definition before the hypernyms. As observed in Subsection 10.3.1, using the entity markers tends to outperform not using special tokens or using focus characters.

In Table 10.4, we report the results on the test set of our two best performing AB4TSV, namely the **swap+em** and **swap+fc**, as well as the best AB4TSV$_{pi}$ model, **default+em**. AB4TSV and AB4TSV$_{pi}$ both outperform the previously reported approaches on the WiC-TSV benchmark. The **swap+em** input format for AB4TSV achieves significantly lower performance than its counterparts on both domain-specific and general instances, and does not generalize well on the test set. Interestingly, on test set and by contrast with the development set AB4TSV and AB4TSV$_{pi}$ achieve similar performance. As the input formatting and analogical formulations are chosen on the development set, the AB4TSV model selected for the benchmark might be over-fitting the development set, while AB4TSV$_{pi}$ generalizes well despite slightly lower performance on the development set.

10.3.3 Performance with regards to analogical data augmentation

We investigate the behavior of AB4TSV with regards to permutations of the AP using the best performing formulation for AB4TSV in terms of accuracy.

Model	**Format**	**Dev** *Acc*	**Dev** $F1$	**AP formulation**
AB4TSV	default	74.5 ± 0.015	77.0 ± 0.016	$cls : descr :: cls : ctx$
	default+fc	74.9 ± 0.010	77.3 ± 0.006	$cls : def :: ctx : cls$
	default+em	75.4 ± 0.027	**77.8** ± 0.023	$tgt : descr :: cls : def$
	swap	75.4 ± 0.016	77.7 ± 0.016	$def : cls :: cls : ctx$
	swap+fc	**75.8** ± 0.013	77.7 ± 0.013	$def : ctx :: cls : hyps$
	swap+em	**75.8** ± 0.017	77.7 ± 0.012	$hyps : def :: cls : ctx$
AB4TSV$_{pi}$	default	74.3 ± 0.016	76.1 ± 0.014	$cls : descr :: cls : ctx$
	default+fc	74.6 ± 0.008	76.6 ± 0.008	$cls : def :: ctx : cls$
	default+em	**75.1** ± 0.014	**77.3** ± 0.013	$tgt : descr :: cls : def$
	swap	74.2 ± 0.010	76.1 ± 0.011	$def : cls :: cls : ctx$
	swap+fc	74.8 ± 0.012	75.9 ± 0.024	$def : ctx :: cls : hyps$
	swap+em	75.0 ± 0.009	76.4 ± 0.011	$hyps : def :: cls : ctx$
		Baselines		
HyperBertCLS	default	74.4 ± 0.014	**77.2** ± 0.009	
	default+fc	73.5 ± 0.027	75.2 ± 0.035	
	default+em	74.0 ± 0.022	76.1 ± 0.019	
	swap	72.6 ± 0.028	74.4 ± 0.031	
	swap+fc	73.1 ± 0.028	75.2 ± 0.031	
	swap+em	**74.6** ± 0.024	76.6 ± 0.022	
HyperBert3	default	74.0 ± 0.014	76.9 ± 0.007	
	default+fc	73.9 ± 0.018	76.3 ± 0.018	
	default+em	73.1 ± 0.031	75.2 ± 0.032	
	swap	73.8 ± 0.015	76.3 ± 0.015	
	swap+fc	73.5 ± 0.011	75.6 ± 0.013	
	swap+em	74.4 ± 0.011	75.7 ± 0.024	

Table 10.3: Accuracy and $F1$-score achieved on the development set by the proposed method and the two baselines. For our model, we report for each input format only the results for the best performing AP formulation.

In Table 10.5, we report the performance of AB4TSV and AB4TSV$_{pi}$ on specific permutations of the input formulation, each model trained for 4 random initializations. As expected, the performance of AB4TSV$_{pi}$ is stable accros permutations when analogical data augmentation is used during training. Conversely, the performance of AB4TSV, which is trained only on the base form of the AP, degrades on the other permutations, in particular for Symmetry of Conformity.

10.4 Conclusion, discussions and perspectives

In conclusion, we successfully reformulated TSV as an analogy detection task, and outperformed competitors on the WiC-TSV benchmark by adapting some elements of the ANN framework, namely ANNc and the analogical data augmentation.

Using analogical data augmentation during training resulted in a model with a more consistent performance across permutations of APs, that generalizes well with comparable performance to a model trained without analogical data augmentation.

Our experiments highlighted the importance of the formatting used as input for BERT, as well as the importance of a suitable analogical formulation to achieve high performance, in particular when not using analogical data augmentation. It would be interesting to confirm whether, when using analogical data augmentation, we observe a similar distribution of the performance across the analogical formulations. However, we leave reproducing the experiment from Subsection 10.3.1 with AB4TSV$_{pi}$ to later work.

The current formulation of AB4TSV is heavily dependent on the formatting of the input, including the order of the elements of the AP used for analogy detection. In Part II, the embeddings manipulated were computed independently from one another. A similar approach might be beneficial for TSV, for example, by splitting the context, definition and hypernyms separately into

Approach	**Accuracy**	**F1**
Supervised		
CTLR [MPD21]	78.3	78.5
[VSD21]	71.9	76.2
BERT-B [Bre+21]	76.6	78.2
BERT-L [Bre+21]	76.3	77.8
FastText [Bre+21]	53.4	63.4
AB4TSV+swap+em	75.7	77.5
AB4TSV+swap+fc	**78.6**	**79.8**
AB4TSV$_{pi}$+default+em	**78.6**	79.4
Unsupervised		
U-dBERT [Bre+21]	61.2	51.3
U-BERT [Bre+21]	60.5	51.9
MIRRORWIC [Liu+21]	73.7	–

Table 10.4: Test set performance of the best performing AB4TSV and AB4TSV$_{pi}$ compared to previously reported results. All results are calculated by the authors of the WiC-TSV benchmark.

Permutation	Model	Accuracy	F1
Base form	AB4TSV	76.2 ± 1.927	78.0 ± 1.932
	AB4TSV$_{pi}$	75.1 ± 1.611	76.8 ± 1.891
Symmetry of Conformity	AB4TSV	53.2 ± 17.10	61.4 ± 18.53
	AB4TSV$_{pi}$	74.5 ± 1.949	76.4 ± 2.210
Exchange of the Means	AB4TSV	72.9 ± 3.596	73.4 ± 5.760
	AB4TSV$_{pi}$	74.7 ± 2.104	76.5 ± 2.315

Table 10.5: Performance on specific permutations of the AB4TSV and AB4TSV$_{pi}$, on the development set, for the **swap+fc** input format and the AP $def : ctx :: cls : hyps$.

BERT.

One important difficulty we encountered was the choice and arrangement of the elements to include in the AP for analogy detection. In Part II, and later in Chapter 11, the elements manipulated were of the same nature and semantic level, while for TSV we considered sentences, words in contexts, and lists of hypernyms. Different formulations of TSV in terms of analogies are possible, but we expect to have similar issues as we had, and potentially larger inputs if we compare multiple TSV instances in the AP, as we did in Chapter 11. Additionally, it would be interesting to extend the work from analogy detection to analogy solving, though an analogy solving formulation appears more suiatble for WSD than for TSV.

Chapter 11

Extension of the ANN framework to FrameNet

Chapter contents

In this chapter, we adapt the ANN framework developed in Part II in the context of frame semantics, focusing on the problem of Frame Semantic Role Labeling (FSRL) on FrameNet-1.7 (FN1.7) (the latest version of FrameNet (FN) at the time of writing). We reformulate FSRL as an analogy solving problem in Section 11.2. Our experiments reported in Section 11.4 show that, under certain conditions, using analogy solving we can obtain results that outperform the SotA approaches on FSRL, without using sophisticated and computationally expensive encoding or decoding mechanisms.

The content of this chapter has not been published at the time of writing.

11.1 FrameNet and frame semantics

We use FN1.7 [Bak17] as our testbed, which essentially provides a lexicon of semantic frames as well as a set of sentences annotated with semantic frames information from this resource.

A semantic frame is a schematic representation of an event or state, which is *triggered* in the sentence by a specific word or expression called the predicate. Each semantic frame contains a set of Frame Elements (FEs), which can core Frame Elements (core FEs) or peripheral/extra-thematic Frame Elements (non-core FEs), that correspond to “various participants, props and other conceptual roles.[1]” We say that a FE is *instantiated* in a sentence by a group of words if the group of words carries the meaning of the FE. In this chapter, we use the notion of Semantic Role (SR) to designate both the FEs and the predicate.

Remark 11.1

The distinction between core FEs and non-core FEs corresponds to whether they are mandatory to express the meaning of the semantic frame or not: a core FEs must be expressed for the semantic frame to be understood while non-core FEs are not necessary. Following Example 11.1 below, the “disembarking” frame can not be understood without knowing the “traveler” and the “vehicle”, which are the core FEs of “disembarking”. However, it is not necessary to know the “manner” in which one disembarks (fast, slowly, carefully, *etc.*) to understand what is described.

Nevertheless, as explained in [Rup+16, Subsection 3.2.3], in practice some core FEs are not expressed in the annotation as they are implied by the context. Therefore, it is safer to consider that any FE of a semantic frame might be omitted.

Frame semantic parsing is the task of obtaining frame semantics annotations from a sentence. It is usually divided in 3 sequential tasks:

1. **predicate identification**: identify all predicates, *i.e.*, words or expressions that trigger a semantic frame;
2. **frame identification**: identify the semantic frames that are triggered by the predicates;
3. **FSRL**: for each semantic frame f and set FE_f of all possible FEs (or *arguments*) defined in f, associate a text span of the sentence with each FE, if such an association exists; FSRL is also known as argument identification and classification, as it sometimes separated into:
   (a) *argument identification*: identifying spans of text that are suitable to be associated with an FE;
   (b) *argument classification*: identifying the FEs corresponding to each span identified in the previous step.

It is important to know that a single sentence may trigger a one or more semantic frames, as in the example Example 11.1.

Example 11.1: Annotation of a sentence with frame semantics

The following sentence was taken from the FSRL test data of FN1.7. It triggers two semantic frames: “posture” annotated in (11.1) below, and “disembarking” annotated in (11.2).

$$\underset{\text{agent}}{\text{[Steve]}}\ ,\ \underset{\text{agent}}{\text{[who]}}\ \text{was}\ \underset{\text{predicate}}{\text{[sitting]}}\ \underset{\text{location}}{\text{[next to John]}}\ ,\ \text{got down in Rome .} \tag{11.1}$$

$$\underset{\text{traveller}}{\text{[Steve , who was sitting next to John ,]}}\ \underset{\text{predicate}}{\text{[got]}}\ \underset{\text{vehicle}}{\text{[down]}}\ \underset{\text{place}}{\text{[in Rome]}}\ . \tag{11.2}$$

As we can see, it is possible to have a group of words involved in multiple annotations, for different semantic frames.

Relations between semantic frames. In the FN ontology, semantic frames are related to each other. These relations are directed, from a super-frame, which is more abstract or less dependent, to a sub-frame, which is more specific or more dependant [Rup+16]. These relations allow to link some SRs of the two semantic frames. For instance, a semantic frame can inherit from another

[1]https://framenet.icsi.berkeley.edu/glossary

semantic frame, which means that it is a more specific version of the semantic frame. As such, the SRs of the super-frame are inherited, and sometimes more accurately specified, by the sub-frame. Following the example from Example 11.1, the “posture” semantic frame, defined by “an Agent supports their body in a particular Location”, inherits from the “state” frame, defined by “an Entity persists in a stable situation called a State.” In this inheritance, the “agent” is a specific kind of sentient “entity”, while the “location” defines a part of the “state”.

Other inter-semantic frame relations exist in the FN ontology than inheritance. Using examples from the Natural Language Toolkit (NLTK) library’s description of FN[2], some important ones are:

- inheritance relations (that we already described);
- usage relations: the sub-frame frame presupposes the super-frame as background, *e.g.*, “speed” uses “motion”;
- sub-frame relations: the sub-frame is a sub-event represented by the super-frame, *e.g.*, “criminal_process” has “arrest” and “trial” among its sub-frames.

In our experiments, we distinguish leaf semantic frames in the inheritance hierarchy, *i.e.*, semantic frames not having any semantic frame inheriting from them in the FN1.7 ontology.

SotA approaches to Frame semantic parsing. In order to perform full frame semantic parsing, current SotA approaches use sophisticated encodings, such as graph neural representations [LSZ21], or decoding mechanisms, such as semi-Markov Conditional Random Fields (CRFs) [Swa+17].

To the best of our knowledge analogies have not been used in the context of FSRL. Swayamdipta, Thomson, et al. [Swa+17] presented a softmax-margin semi-Markov model. The authors use a bidirectional RNN with a semi-Markov CRF without initially using any syntactic features, and then employ multi-task learning and syntactic scaffolding to obtain SotA results at the time of publication. More recently, Lin, Sun, and Zhang [LSZ21] used Graph Neural Networks (GNNs) based on BERT embeddings and Bidirectionnal Hierarchical Long- and Short-Term Memory neural networks (BiHLSTMs) [SGS15] for the full frame semantic parsing task, also obtain SotA results.

11.2 Analogical transfer for FSRL

In this section, we reformulate the FSRL task as an analogy solving task, and describe the model we used to tackle it.

11.2.1 mBert contextual word embeddings

Contextual information is necessary to understand the SR of a group of words, as it is defined in relation to a semantic frame within a sentence. For instance, in Example 11.1, knowing whether “John” or “Steve” is the agent of semantic frame “posture” depends on the sentence considered. Using “John, who was sitting next to Steve.” instead of the sentence of Example 11.1 would result in “John” being the agent instead of “Steve”.

Accordingly, we decided to focus on contextualized word embedding models. We decided to use a model of the BERT family [Dev+19, see also Subsection 3.3.5] for three main reasons:

- they offer the Extractive Question Answering (Ex-QA) formulation, with ready to use implementations in major frameworks and well-known performance, that we use as a basis for formatting our analogical equations as “analogical questions”;
- BERT models are widely used which allows better comparability with other approaches, for instance Lin, Sun, and Zhang [LSZ21] who also use a BERT model;
- compared to some recent models with more parameters such as GPT4[3] (1.76 trillion parameters) or Mixtral[4] (45 billion parameters), BERT models (108 million parameters) have a lower fine-tuning cost, however this limitation can be mitigated by using model instances of a smaller size but weaker performance.

[2]`https://www.nltk.org/howto/framenet.html`
[3]`https://platform.openai.com/docs/models`
[4]`https://mistral.ai/fr/news/mixtral-of-experts/`

Among the existing variants of BERT, we use mBert [Dev+19] as we intend to expand our application to other languages in future work, as discussed in Subsection 11.5.4. The mBert architecture considers tokens which are different from words in the linguistic meaning, as for instance a word may be split in multiple tokens and tokens can be punctuation marks, as explained in Subsection 3.3.4.

11.2.2 Reformulation of FSRL as analogy solving

As mentioned at the start of this chapter, we focus on predicting SRs through analogy solving. To achieve this, our formulation considers two sentences s, t, and we perform what can be called analogical transfer: by analogy solving, we transfer the knowledge we have in the source sentence s, in our case the SR annotations of some frame f, to the target sentence t. The analogical equation $A : B :: C : x, x = D$ is such that A, B are SRs of f in s, and C, D SRs of f in t. Furthermore, A, C are instances of the same SR r, and B, D instances of the same SR r' (potentially different from r). This analogical equation can be read as "Which group of words in t has the same role *w.r.t.* C, as the role B has *w.r.t.* A in s?"[5] In Example 11.2, we present some examples of such analogical equation and how we can read them.

Example 11.2: Examples of analogical equations between SRs in FN

Taking the example of the semantic frame "activity start"[a] for f, we have the following SRs: the predicate p, as well as the core FEs "agent" and "activity". For the example, we do not consider non-core FEs.

Let us now consider the following annotated sentences:

- $s =$"Most PAT dogs are mature when $\underbrace{\text{[they]}}_{\text{agent}}$ $\underbrace{\text{[commence]}}_{\text{predicate}}$ $\underbrace{\text{[work]}}_{\text{activity}}$."
- $t =$"$\underbrace{\text{[The Labour Party]}}_{\text{agent}}$ did not $\underbrace{\text{[enter]}}_{\text{predicate}}$ $\underbrace{\text{[into negotiations]}}_{\text{activity}}$."

From these sentences we can create the following analogical equation $A : B :: C : x, x = D$, among the 6 permutations of 2 out of the 3 SRs:

- if $r =$ predicate and $r' =$ "agent", then we have:

$$\underbrace{\text{[commence]}}_{\text{predicate}} : \underbrace{\text{[they]}}_{\text{agent}} :: \underbrace{\text{[enter]}}_{\text{predicate}} : x, x = \underbrace{\text{[The Labour Party]}}_{\text{agent}}$$

  which can be read as "What is in t the group of words which has the same role *w.r.t.* $C =$ "enter", as the role $B =$ "they" has *w.r.t.* $A =$ "commence" in s, *i.e.*, the "agent" in the semantic frame where A, C are predicates?"

- if $r =$ "agent" and $r' =$ "activity", then we have:

$$\underbrace{\text{[they]}}_{\text{agent}} : \underbrace{\text{[work]}}_{\text{activity}} :: \underbrace{\text{[The Labour Party]}}_{\text{agent}} : x, \quad x = \underbrace{\text{[into negotiations]}}_{\text{activity}}$$

  which can be read as "What is in t the group of words which has the same role *w.r.t.* $C =$ "The Labour Party", as the role $B =$ "work" has *w.r.t.* $A =$ "they" in s, *i.e.*, the "activity" in the semantic frame where A, C are "agents"?"

[a] https://framenet2.icsi.berkeley.edu/fnReports/data/frame/Activity_start.xml

We split the analogical transfer formulation in two distinct settings:

- the FSRL setting where A, C are the predicates and B, D are FEs;
- a generalization where A, B, C, D are SRs

The FSRL setting is a particular case of the general analogical transfer setting, as predicates and FEs are gathered under the notion of SRs.

[5] This reading can be interpreted in a similar manner as what is done in Question-Answer Driven Semantic Role Labeling (QA-SRL) [HLZ15], where questions are used to find each FE, *e.g.*, "who started something?" to obtain the "agent" of "activity start". In our case, the SR is described using A, B, C.

General formulation. Given a source sentence $s = \{w_1^s, \dots, w_n^s\}$ and a distinct target sentence $t = \{w_1^t, \dots, w_m^t\}$ represented by their sequence of tokens, we will consider three substrings of consecutive tokens in s and t, respectively:

$$\begin{aligned} A &= \{w_{i_A}^s, \dots, w_{i_A+|A|-1}^s\} \quad \text{in} \quad s, \\ B &= \{w_{i_B}^s, \dots, w_{i_B+|B|-1}^s\} \quad \text{in} \quad s, \\ C &= \{w_{i_C}^t, \dots, w_{i_C+|C|-1}^t\} \quad \text{in} \quad t, \end{aligned}$$

with i_A, i_B, i_C representing the indices of the starting token position for A, B, C respectively. With $\mathcal{R}_f$ the SRs of a semantic frame f, A and C instantiate the same SR $r \in \mathcal{R}_f$, respectively in s and t. We seek to identify $D = \{w_{i_D}^t, \dots, w_{j_D}^t\}$ with $i_D, j_D \in [1, m]$ and $i_D \leq j_D$ such that B, D instantiate the same SR $r' \in \mathcal{R}_f$. In other words, we seek D that solves the analogical equation $A : B :: C : x$.

FSRL formulation. In the FSRL task, given a target sentence t, a predicate p_t of t triggering a semantic frame f, we seek to identify all spans of text in the sentence that are associated with a FE $r \in \mathcal{R}_f$. To do so, we select a source sentence s that triggers the same frame f and that has been annotated with all its SRs, and reformulate FSRL into multiple analogical equations.

In practice, we assume that a SotA semantic frame and predicate annotation method has been applied on the target sentence t we want to annotate, providing us with the semantic frame f and the predicate p_t. Relying on these first annotations, we create analogical equations to predict each of the FEs. In each analogical equation, we set r = predicate, and create a variant of the analogical equation for each FE r' instantiated in s. The annotation for each FE in t is the solution to the corresponding equation, found by analogy solving. An example of this process is given in Example 11.3.

Example 11.3: Examples of analogical equations for FSRL

Let us take once again the sentence t:

$$t = \text{“The Labour Party did not } \underbrace{\text{[enter]}}_{\text{predicate}} \text{ into negotiations.”}$$

for which we know the predicate p_t = “enter” and the semantic frame f = “activity start”, using a SotA semantic frame and predicate annotation tool.

We use the source sentence:

$$s = \text{“Most PAT dogs are mature when } \underbrace{\text{[they]}}_{\text{agent}} \ \underbrace{\text{[commence]}}_{\text{predicate}} \ \underbrace{\text{[work]}}_{\text{activity}} \text{ .”}$$

in which we have the FEs “agent” and “activity” annotated. Therefore, we can create the two following analogical equations:

- $s, t, \underbrace{\text{[commence]}}_{\text{predicate}} : \underbrace{\text{[they]}}_{\text{agent}} :: \underbrace{\text{[enter]}}_{\text{predicate}} : x$, which once solved gives us the annotation $x = \underbrace{\text{[The Labour Party]}}_{\text{agent}}$;
- $s, t, \underbrace{\text{[commence]}}_{\text{predicate}} : \underbrace{\text{[work]}}_{\text{activity}} :: \underbrace{\text{[enter]}}_{\text{predicate}} : x$, which once solved gives us the annotation $x = \underbrace{\text{[into negotiations]}}_{\text{activity}}$.

Remark 11.2

Our approach could be extended by using the prediction of each SR to improve and cross-check the predictions on the other SRs, in a similar form as the last analogical equation of Example 11.2.

More precisely, once we have solved the analogical equations where A, C are fixed as predicates, we could create new analogical equations where A, C would be FEs using the previous results to specify C. Doing so, if we manage to reproduce the annotations using different FEs for A, C this would increase the confidence in the predictions, while obtaining different annotations for an FE would indicate that further verifications are necessary.

11.2.3 Analogical Model Formalization

We define two probability distributions $\mathbf{p}_b, \mathbf{p}_e$ over the tokens of t, respectively the likelihood of a token being the first token of the answer (the **b**eginning) and the last token of the answer (the **e**nd). The two probability distributions are conditioned by s, t as well as by the analogical equation $A : B :: C : x$ to solve. Hence, the analogy solving problem can be formulated as follows: for $i_D \leq j_D$,

$$\begin{aligned} i_D &= \underset{i \in [0,m]}{\operatorname{argmax}} \, \mathbf{p}_b(w_i^t | s, t, A, B, C), \\ j_D &= \underset{j \in [0,m]}{\operatorname{argmax}} \, \mathbf{p}_e(w_j^t | s, t, A, B, C). \end{aligned} \tag{11.3}$$

For each token, the conditional probabilities $\mathbf{p}_b, \mathbf{p}_e$ of being the start or end of fourth element of an analogy, given the two sentences and the first three elements of the analogy, are obtained using mBert.

Ex-QA formulation. To obtain these probabilities, we use the *Ex-QA*[6] model for mBert, proposed for solving the Stanford Question Answering Dataset (SQuAD) [Raj+16]. For each token $w_i \in t$, we obtain contextual embeddings $\mathbf{w}_i = \text{mBert}(w_i | s, t, A, B, C)$ which we then feed to two perceptrons that learn whether a token constitutes the beginning or end of the solution to the analogical equation. More precisely, we estimate

$$\begin{aligned} \mathbf{z_b}(i) &= \mathbf{W}_b^T \mathbf{w}_i + \mathbf{b_b}, \\ \mathbf{z_e}(i) &= \mathbf{W}_e^T \mathbf{w}_i + \mathbf{b_e}, \end{aligned}$$

where $\mathbf{W}_b, \mathbf{W}_e$ are learned weights and $\mathbf{b}_b, \mathbf{b}_e$ learned biasses of the perceptrons. Conditional probabilities are obtained for each token given the context using a softmax function:

$$\begin{aligned} \mathbf{p}_b(w_i^t | s, t, A, B, C) &= \frac{e^{\mathbf{z}_b(i)}}{\sum_{j \in t} e^{\mathbf{z}_b(j)}}, \\ \mathbf{p}_e(w_i^t | s, t, A, B, C) &= \frac{e^{\mathbf{z}_e(i)}}{\sum_{j \in t} e^{\mathbf{z}_e(j)}}. \end{aligned}$$

Notice that in Equation (11.3) it is possible to have $i, j = 0$. Inspired by [Dev+19], we consider a special token w_0^t that helps us handle instances in which no solution exists. This is the case when the optimal solution for Equation (11.3) yields $i = j = 0$, denoting a negative instance (as detailed in Subsection 11.3.1). Otherwise, we consider only $0 < i \leq j$ during decoding.

11.2.4 Non-analogical transfer model

We define an additional model, used in the ablation study in Subsection 11.3.3 to confirm the benefit on the performance of the analogical formulation including A, C. It is in all points identical to the one defined in Subsection 11.2.3, except that A, C do not appear in the input. The optimization problem becomes:

$$\begin{aligned} i_D &= \underset{i \in [0,m]}{\operatorname{argmax}} \, \mathbf{p}'_b(w_i^t | s, t, B), \\ j_D &= \underset{j \in [0,m]}{\operatorname{argmax}} \, \mathbf{p}'_e(w_j^t | s, t, B), \end{aligned} \tag{11.4}$$

where $i_D \leq j_D$, and $\mathbf{p}'_b, \mathbf{p}'_e$ correspond to $\mathbf{p}_b, \mathbf{p}_e$ without A, C. This can be seen as a simple transfer of r' from s to t, instead of the analogical transfer we perform with the main model.

[6]https://huggingface.co/docs/transformers/main/en/model_doc/bert#transformers.BertForQuestionAnswering

Model	Context s	Context $A : B :: C : x$ or B	Question t
Analogical	`[CLS]` `[s]` $w_1^s \dots w_n^s$	`[A]` A `[B]` B `[C]` C	`[SEP]` `[t]` $w_1^t \dots w_m^t$ `[SEP]`
Non-analogical	`[CLS]` `[s]` $w_1^s \dots w_n^s$	`[B]` B	`[SEP]` `[t]` $w_1^t \dots w_m^t$ `[SEP]`

Table 11.1: Input format of the mBert Ex-QA model for the analogical (described in Subsection 11.2.3) and non-analogical (described in Subsection 11.2.4).

11.2.5 Input format for mBert.

For the analogical and the non-analogical models, our approach extends on the Ex-QA input format implemented in the HuggingFace library: "`[CLS] question [SEP] context [SEP]`", where `[CLS]` and `[SEP]` are special tokens defined by mBert. The *context*, which specifies where the answer should be found, corresponds to t in our task. The *question* conditions the (semantic) content of the answer, and corresponds to $s, t, A : B :: C : x$ in our case. However, it is not necessary to provide t in both the context and the question, so we limit the question to $s, A : B :: C : x$. To indicate the boundaries of each element of our formulation to the transformer model, we add our own special tokens: `[s]`, `[t]`, `[A]`, `[B]`, and `[C]`. This results in sequences following the pattern in Subsection 11.2.5. Reusing the example from Example 11.2, we obtain the inputs in Example 11.4.

Example 11.4: Example analogical and non-analogical inputs for mBert

Let us consider the following sentences and analogical equation:

$$s = \text{Most PAT dogs are mature when } \underset{\text{agent}}{[\text{they}]}\ \underset{\text{predicate}}{[\text{commence}]}\ \underset{\text{activity}}{[\text{work}]}\ .$$

$$t = \underset{\text{agent}}{[\text{The Labour Party}]}\ \text{did not}\ \underset{\text{predicate}}{[\text{enter}]}\ \underset{\text{activity}}{[\text{into negotiations}]}\ .$$

$$\underset{\text{agent}}{[\text{they}]} : \underset{\text{activity}}{[\text{work}]} :: \underset{\text{agent}}{[\text{The Labour Party}]} : x, \quad x = \underset{\text{activity}}{[\text{into negotiations}]}$$

The input for the analogical model (see Subsection 11.2.3) will be:

`[CLS]` `[s]` Most PAT dogs are mature when they commence work . `[A]` they `[B]` work `[C]` The Labour Party `[SEP]` `[t]` The Labour Party did not enter into negotiations . `[SEP]`

The input for the non-analogical model (see Subsection 11.2.3) will be:

`[CLS]` `[s]` Most PAT dogs are mature when they commence work . `[B]` work `[SEP]` `[t]` The Labour Party did not enter into negotiations . `[SEP]`

Finally, the expected output would be:

Token position	0	1	2	3	4	5	6	7	8	9
Token	∅	The	Labour	Party	did	not	enter	into	negotiations	.
Expected p_b	0	0	0	0	0	0	0	**1**	0	0
Expected p_e	0	0	0	0	0	0	0	0	**1**	0

11.3 Analogy solving performance

Following the formulation introduced in Section 11.2, we train an analogy solving model on the training data described in Subsection 11.3.1. We determine the limitations of our model with regards to the analogical setting, and the conclusions drawn here can be transferred to the FSRL setting. Namely, our FSRL formulation is a special case of the general analogical formulation where A, C are limited to predicates.

Task	Data subset	Instances Positive	Instances Negative
Analogical transfer	Training set	2.751 ± 0.645	2.623 ± 0.655
	Development set	2.972 ± 0.695	2.488 ± 0.594
	Test set	2.737 ± 0.634	2.428 ± 0.576
	$r_A \neq r_C$	3.179 ± 0.393	
	Frame mismatch	2.947 ± 0.688	
FSRL	Training set	1.822 ± 0.793	
	Test set	1.816 ± 0.750	

Table 11.2: FEs per semantic frame and per sentence (excluding the predicate), for all the experiments.

11.3.1 Experimental setup

To train our model and explore its analogy solving performance, we extract sentence examples from the FN1.7 ontology to build APs $A : B :: C : D$, where A, B, C, D are either instances of core FEs or the predicate. For each frame, we gather up to 1000 sentences with the annotation status of either "FN1_Sent", "Finished_Initial", or "Finished_Checked", the 3 annotation status of the highest quality according to the documentation of FN1.7.

As mentioned in Subsection 11.2.3, it is possible that some SRs of a given frame are not instantiated in a given sentence, in particular for non-core FE which are optional to the meaning of the semantic frame. To account for this, we consider positive instances of the analogical equation that can be solved because r' is instantiated in t as D, and negative instances where r' is not instantiated in t and the analogical equation cannot be solved.

Data augmentation. To integrate analogical knowledge in our model, we use a data augmentation process based on Symmetry of Conformity and Exchange of the Means, similar to what was done in Part II (see Subsection 6.3.1). For each pair of SRs r, r' of a semantic frame and each pair of sentences s, t instanciating the frame, we generate the 8 equivalent APs: $A : B :: C : x,\ x = D$, $A : C :: B : x,\ x = D$, $D : B :: C : x,\ x = A$, $C : A :: D : x,\ x = B$, $C : D :: A : x,\ x = B$, $B : A :: D : x,\ x = C$, $D : C :: B : x,\ x = A$, $B : D :: A : x,\ x = C$. We exchange s with t as needed when we perform these permutations, such that the solution to the equation appears in the *context* of the Ex-QA formulation. We exclude from our study APs where $A = B$ and $C = D$. Indeed, the corresponding analogical equations would become $A : A :: C : x$ and the solution $x = C$ can be found without needing to explore the semantic relations between the elements of the analogical equation, by Identity. Such trivial examples could degrade the quality of the training of the model.

Dataset size. To maintain a good balance in the SRs presented, we sample APs such that we have the same amount from all possible pairs of SRs of each pair of sentences.

To make our training set and development set, we select randomly 250 semantic frames from the leaf semantic frames (see Section 11.1). Similarly, we select another 100 leaf semantic frames used for both the analogical test set and the $r_A \neq r_C$ set used in Subsection 11.3.3.

For our training set, we take up to 1000 positive and 1000 negative instances per semantic frame. From them, we take out 1000 positive and 1000 negative instances to make the development set, without considering which frame instances are from. In total, the training set contains 249000 positive and 199816 negative instances, and the development set contains 1000 positive and 1000 negative instances. For the test set, we take 100 instances of each class for each semantic frame, for a total of 10000 positive and 8030 negative instances.

The distribution of FEs per semantic frame and per sentence is summarized in Table 11.2.

Training hyperparameters. The model (including mBert) is trained for at most 1 epoch. Batch size is automatically found by the HuggingFace library to maximize Graphics Processing Unit (GPU) usage. Early stopping is decided on the development set, using an approximation of the Word Error Rate (WER) that uses the tokens' positions instead of words. We approximate the

Model	Instances	Accuracy	Wrong SR	Not an SR	SR not found
Analogical model (using A,B,C)	Positive	**72.31%**	0.28%	**11.24%**	**16.17%**
	Negative	72.09%	0.52%	27.38%	—
	All	**72.21%**	0.39%	**18.43%**	**8.97%**
	$r_A \neq r_C$	70.75%	0.31%	11.59%	17.36%
Non-analogical model (using only B)	Positive	53.46%	**0.10%**	16.59%	29.85%
	Negative	**75.32%**	**0.39%**	**24.30%**	—
	All	63.19%	**0.23%**	20.02%	16.56%

Table 11.3: Analogy solving results (in % of all instances) for the analogical and non-analogical models. The boldface numbers correspond to best performance (*i.e.*, the highest for accuracy and the lowest for the various types of errors) between the two models. Instances where $r_A \neq r_C$ are not counted in the overall performance (*All*), nor in the boldface numbers.

number of editions of the WER with the number of positions which are not shared by the prediction and the expected output. It is noted $|\{i^\star_D, \ldots, j^\star_D\}\Delta\{i_D, \ldots, j_D\}|$ with Δ the symmetric difference, i_D, j_D the predicted and $i^\star_D, j^\star_D$ the expected beginning and end positions for D. This result in the following formula:

$$\mathrm{PosWER}(i_D, j_D, i^\star_D, j^\star_D) = \frac{|\{i^\star_D, \ldots, j^\star_D\}\Delta\{i_D, \ldots, j_D\}|}{j^\star_D - i^\star_D + 1}.$$

Evaluation method. For all instance classes, we report the accuracy of the model, which is the percentage of instances where the model returns the expected output: the *gold SR* for positive instances, and the w^t_0 token for negative instances.

If the model does not return the expected output, we speak of model failure and consider 3 possibilities:

- "*wrong SR*" if the model returns a instance of an SR that is different from the gold SR;
- "*SR not found*" if the model outputs w^t_0 even if the analogy could be solved (*i.e.*, positive instances);
- any other case corresponds to outputs that do not exactly match an SR nor the w^t_0 span, that we call "*not an SR*".

Note that the above three situations cover all the possible cases of model failure, so *accuracy* + *wrong SR* + *SR not found* + *not an SR* = 100%.

11.3.2 Results

We report in Table 11.3 the performance of our model on positive instances (solvable analogical equations) and negative instances (unsolvable analogical equations due to missing SR instance), as well as the average performance over those two classes of instances. *SR not found* is not given for negative instances, as it is the expected output.

We also report in the same table the results for several ablation experiments reported in Subsection 11.3.3.

Overall performance. There is no significant difference between the accuracy on positive and negative instances, with a high level of performance (above 72% accuracy) in both cases. Overall, our model has a high accuracy, despite the punitive way we determine failures: in the case of multi-token words, the model fails if a token part of a word is omitted while the other tokens of the word are correctly predicted and, conversely, for tokens that are wrongly predicted.

Analysis of error rates. With positive instances, the model wrongly determines that the SR r' is not instantiated in only about 16% of cases. However, for negative instances, the model errors are almost exclusively *not an SR*.

As missing annotations and annotations errors are present in the part of FN1.7 we use, it is likely that some of our instances are solvable but the instance for r' is not labeled, so the instances are counted as negative ones. Although we did not find the exact inter-annotator agreement for FN1.7, for a similar frame semantics annotation task in French, Djemaa, Candito, et al. [Dje+16] report 77% inter-annotator agreement for SRs of matching semantic frames.

11.3.3 Ablation study on the impact of A, C

We study the sensitivity of the model to several perturbations regarding the SRs A, C for two purposes: we measure the impact of A, C on the performance of the model from the analogical point of view, and, by extension, the impact of errors in identifying the predicate on the FSRL performance.

To do so, in a first experiment we generate analogies such that A, C instantiate different roles $r_A \neq r_C \in \mathcal{R}_f$ but B, D instantiate the same role r'. While the analogy is erroneous from the point of view of APs, it is possible to solve it by transferring r' from s to t. In a second experiment, in a setting in all other aspects equivalent to the analogy solving model we propose, we remove A, C, resulting in the model defined in Subsection 11.2.4. Once again, this setting is a transfer of r' from s to t, however this time the model cannot rely on A, C to identify the relevant semantic frame.

Introducing a mismatch between the SRs instantiated by A and C. We take 100 positive and 100 negative instances of the for each frame, for a total of 7760 instances with $r_A \neq r_C$. These instances are generated by taking, for each pair of sentences of a frame, triplets of distinct SRs r_A, r_C, r' such that r' is instantiated in both sentences, r_A is instantiated at least in the first sentence, and r_C is instantiated at least in the second sentence. Therefore, it is possible, but not mandatory, that r_C is not instantiated in s or r_A not instantiated in t. We test the model trained in Section 11.3 on this new data, and results are shown in the $r_A \neq r_C$ row of Table 11.3.

While there is a drop in accuracy in this setting, the performance remains very high, with only a 2% decrease. Additionally, it is interesting to see that the new errors mostly belong to the *SR not found* category. While the difference might not be significant enough to draw conclusions, we propose the following hypothesis: by introducing a mismatch $r_A \neq r_C$ in the starting point of the relation, the model determines that there is no instance that would fit closely enough the erroneous relation r_A to r' when starting from C. Despite this potential negative effect, using A, C likely help the model better identify the meaning of the frame in t.

Remark 11.3

The $r_A \neq r_C$ setting is a form of analogy, although weaker than the standard setting (with $r_A = r_C$) as instead of having the AP:

$$\text{“}r\text{ in }s\text{”} : \text{“}r'\text{ in }s\text{”} :: \text{“}r\text{ in }t\text{”} : \text{“}r'\text{ in }t\text{”},$$

we have:

$$\text{“some SR of }f\text{ in }s\text{”} : \text{“}r'\text{ in }s\text{”} :: \text{“some SR of }f\text{ in }t\text{”} : \text{“}r'\text{ in }t\text{”}.$$

Following the distinction from Barbot, Miclet, and Prade [BMP19, see also Section 2.1], the standard setting is an AP, while $r_A \neq r_C$ is a relational proportion, a different form of analogy where some permutations are less relevant.

Removing A, C from the input. To confirm the benefit of A, C on the performance, we use the model defined in Subsection 11.2.4, which can be seen as a simple transfer of r' from s to t instead of the analogical transfer we perform with the main model. We train this model on the same data as before, however we discard any information on A and C. The performance of this new model on the test data used for the analogical models is reported in the first three rows of Table 11.3.

We observe a significant drop in performance close to 19% for positive instances, with most of this gap transferred to *SR not found*. Following the more frequent *SR not found* answers, the accuracy on negative instances improves, but only by 3%, resulting in a 9% drop in overall performance when we do not use analogical transfer.

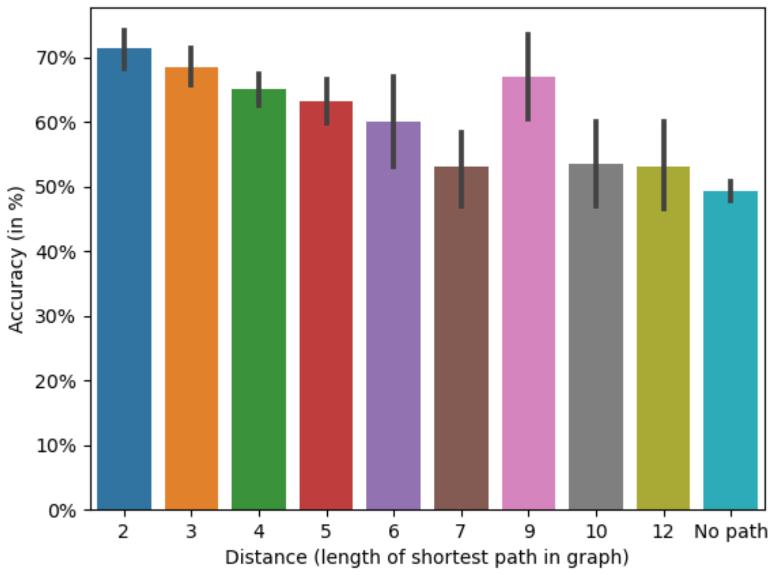


Figure 11.1: SPL against analogical model accuracy for *Inheritance* relations. Error bars are the 95% confidence interval.

11.3.4 Ablation study on the relatedness of semantic frames used for s and t.

In our problem formulation, we state that s, t activate the same semantic frame. In this subsection, we report ablation experiments we performed when the previous condition is not respected and s, t trigger different semantic frames. Our starting intuition is that, as our model relies on semantic relations, if the semantic frames of s, t are different but semantically related, we should maintain high analogy solving performance. More specifically, the semantically closer the semantic frames are, the higher the performance we should obtain, because semantically close semantic frames have similar SR structure. For instance, the semantic frames "Process_end"[7] and "Process_stop"[8] are semantically very close, and both have the "Process" core FE, as well as the exact same non-core FEs even if their meaning differ slightly.

Experimental setup. The relations between semantic frames indicated in the ontology of FN1.7 do not cover many semantic frames, with a relation density[9] of the order of magnitude of 10^{-5} for all relations, except for *Inheritance* relations (called *Inherits from* and *Is Inherited by* in FN1.7 [Bak17; BFL98]) which is closer to 10^{-4}.

To compute how related two semantic frames are, we compute the smallest number of steps to reach one from the other following the relation. This corresponds to the node distance in the undirected graph of each relation, *i.e.*, the Shortest Path Length (SPL).

We considered 100 randomly selected pairs of semantic frames, and for each of them we considered core FEs that are labeled the same in the two semantic frames as semantically close enough to create an analogical equation. For each semantic frame pair, we generate up to 100 (positive) instances. In particular, for *Inheritance*, we obtained a total of 3528 instances with SPL ranging from 2 to 12, as well as 4506 instances involving unrelated frame pairs, for a total of 9834 instances.

Results. We were able to identify a tendency matching our intuition for *Inheritance* relations, as we observe a correlation between model performance and relatedness in terms of SPL, with a very significant Spearman correlation p-value $= 5.18e - 68$ (Spearman correlation coefficient $\rho = -0.1744$). This is further supported by the clear trend of the performance we observe in Figure 11.1, where we report the performance for each SPL value. For frames that are closely related, the performance is almost the same as when the sentences trigger the same frame (71.22%

[7]`https://framenet2.icsi.berkeley.edu/fnReports/data/frame/Process_end.xml`

[8]`https://framenet2.icsi.berkeley.edu/fnReports/data/frame/Process_stop.xml`

[9]The density of a relation between semantic frames is number of pairs of semantic frames that are related divided by the total number of frame pairs.

for a SPL of 2 against 72.21% when the frame is the same, and 49.49% for unrelated frames). These results indicate a certain tolerance of our approach with regards to the frame instantiated in the source sentence, which can help mitigate the scarcity of labelled data for some frames.

11.4 FSRL performance

At the end of Subsection 11.2.2, we introduced a procedure to annotate each FE of unseen sentences using analogical transfer.

In this section, we perform several experiments and discuss the limitations of the approach described above. To demonstrate the feasibility of this approach, we apply our method on FN1.7. In particular, under ideal conditions, we are able to compete with the ideal performance of the model with the SotA in the same setting. We also discuss some challenges of the selection of the source sentence s, the main limiting factor of our approach.

11.4.1 Experimental setup

In this experiment, we use the analogical model trained in Section 11.3.

Labelling multiple FEs using the same source. When implementing the approach, a key concern is the selection of the source sentence. We use two sentence selection settings:

1. we use potentially different sources for each FE;

2. we use the same source sentence for all the FEs of the frame.

An FE of the test set is not covered if none of the sentences in the training set activate the corresponding semantic frame or if the SR is never instantiated for this frame in the training sentences. For instance, a source s could instantiate a non-core FE r' but not instantiate another non-core FE r'', while conversely a source s' would instantiate r'' but not r'. Overall, setting 1. allows to predict all the FEs but requires suitable retrieval of a source for each of them, while setting 2. minimizes the retrieval effort at the cost of uncovered FEs.

Source sentence selection. In our experiments, we perform analogical transfer for all possible sources, and perform source selection a posteriori, for approximately a million of analogical equations.

To determine the upper limit of the performance of the model, we select the best possible source in each setting. In setting 1., we take for each frame element any sentence that allows a successful prediction, while in setting 2. we take the sentence with the highest accuracy on the current frame. A similar process is used to select the worst and obtain the lower limit of the performance of our model. To obtain a first estimate of the performance of the model in a realistic setting, we average the accuracy over all possible sources for each FE. This simulates a random selection algorithm, *i.e.*, a naive approach that assumes we have no criterion on how to select an appropriate source other than the semantic frame it activates. More involved source selection processes are explored in Subsection 11.4.3.

Dataset. To fit the FSRL task used in the literature, we apply our method on the test set of FN1.7. For the source sentences, we use the corresponding training set. In this experiment, we consider not only core FEs but also non-core FEs, contrary to the general analogical setting in Section 11.3. For some semantic frames in the test set, some FEs do not appear in the training set. In particular, with setting 1., 95.25% of all FEs are covered, and 93.13% of semantic frames have all their FEs covered. With setting 2., at best 88.24% of all FEs and 91.27% of semantic frames are fully covered. The distribution of FEs per semantic frame and per sentence for FSRL is summarized in Table 11.2.

Performance. To measure the performance of our model in a manner comparable with the SotA approaches, we use the F1. As mentioned above, settings 1. and 2. do not cover all the FEs in the

Source for the FEs	Setting 1. Different	Setting 2. Same
Worst source	9.59%	14.91%
Random source	47.99%	42.57%
Best source	**76.08%**	**73.40%**
Best from [LSZ21]	72.22%	

Table 11.4: FSRL F1 given the gold semantic frame and predicate, on the FN1.7 test set.

	Frame seen in model training?	
	No	Yes
Non-core FE	65.71% (1630)	66.47% (170)
Core FE	80.22% (6890)	81.12% (2076)

Table 11.5: FSRL accuracy on covered FEs, given the gold semantic frame and predicate. Only setting 1. using the best source is reported. The number of FEs in each category is reported in parenthesis.

test set. This is taken into account in the way we compute precision and recall for F1:

$$precision = \frac{\#\text{successfully predicted SRs}}{\#\text{covered SRs}},$$
$$recall = \frac{\#\text{successfully predicted SRs}}{\#\text{covered SRs} + \#\text{not covered SRs}},$$
$$F1 = \frac{2}{precision^{-1} + recall^{-1}}.$$

11.4.2 Performance

In Table 11.4, we compare the performance of our approach with the FSRL results in [LSZ21], which contains the SotA on FN1.7 to the best of our knowledge. We limit ourselves to the FSRL setting where the gold semantic frame and predicate is given.

Performance under the best conditions. As mentioned above, the basic use case for our approach is to label FEs one by one and independently. Setting 1. corresponds to this approach, where we are able to use the most fitting source for each FE. In this case, our model outperforms the best model from [LSZ21] by a little under 4% under the best conditions.

Setting 2. explores what happens when only one sentence is presented to the system, and all the available FEs are transferred using our analogical model. This restriction has the advantage of reducing the number of sources to retrieve from the base of sources. Interestingly, under the best conditions performance drops by less than 3% compared to setting 1. despite the uncovered FEs, and setting 2. outperforms the SotA model from [LSZ21] by around 1%.

Performance beyond the best conditions. Using random source selection, the model performance drops under 50% in both settings, and below 15% for the worst source. The performance gap between the best, random, and worst source highlights the importance of a sound source selection process. We performed preliminary experiments with sentence embedding models to get further insight on those results, reported in Subsection 11.4.3.

Performance for core and non-core FEs. Our analogical training set was directly built from the FN1.7 ontology, without accounting for the training set / test set split used in FSRL. Our training set covers only core FEs of a subset of all semantic frames, corresponding to 22.38% of the semantic frames of the FSRL test set. In Table 11.5, we report the performance and the number of FEs depending on whether the corresponding semantic frame is in the analogical training data and whether it is a core FE.

We notice a significant drop in performance between core FEs and non-core FEs of around 15%, which is expected as only core FEs were seen in training. The semantic link between non-core FEs and the predicate is more subtle than between core FEs and the predicate, which could explain such a significant drop in performance. We also notice that performance does not differ significantly between semantic frames seen during training and those unseen, highlighting the ability of our approach to generalize to unseen semantic frames.

11.4.3 Preliminary experiments on source sentence selection

It appears clear that, in order to achieve the full potential of analogical transfer, we need to further investigate the source sentence selection mechanism.

Source selection with sentence embedding similarity. In this direction, we performed preliminary experiments using out-of-the-box sentence similarity approaches on sentence embeddings. The models we used include MiniLM [Wan+20a] and two variants of MPNet [Son+20][10], as well as the mBert model we fine-tuned for analogy solving. We did not fine-tune the sentence embedding models for finding the most suited sources, which would likely significantly improve performance.

To find the source, for all models we compared the embedding of the target sentence with the potential sources using the dot product, cosine similarity, as well as Euclidean and Manahalobis distances, inspired by Wijesiriwardene, Wickramarachchi, et al. [Wij+23].

Performance. Overall, the performance does not differ significantly between the sentence embedding models we tested, with an F1 ranging from 44% to 47% in setting 2. While the results obtained are only slightly higher than *random* selection, using sentence embeddings to retrieve the most appropriate sources appears promising. Indeed, considering MiniLM and the dot product score, only a small portion of target sentences have a very similar source available (1.34% of FEs above 0.7). Taking only these highly similar sources we reach 70.45% of F1, only 2% under the best model from [LSZ21]. We also found a significant correlation between the sentence similarity and the FSRL performance when considering slices of 0.1 on the similarity, with a Pearson correlation p-value $= 6.73e{-5}$ and a correlation coefficient of $\rho = 0.8997$.

11.5 Conclusion, discussions and perspectives

We adapted analogical transfer for FSRL and showed its potential to outperform SotA results, using a simple model based on Ex-QA. When compared to a model that does not rely on analogical transfer, such as the non-analogical transfer model of our experiments, analogical transfer is capable of dramatically increasing results on identifying SRs without compromising performance when the SR is absent from the target sentence. We go further on those aspects and possible improvements in Subsection 11.5.1. Nevertheless, as discussed in Subsection 11.5.2, the mechanism of source sentence selection was identified in our experiments as being of key interest to achieve good analogical transfer performance. Therefore, in future work we plan to focus on the problem of sentence selection, exploring more sophisticated models to better leverage the semantic embedding space.

Our research also resulted in an analogical dataset that complements traditional datasets of analogies between words on factual and lexical semantics [DGM16], discussed in Subsection 11.5.3. We envision extensions to other languages, as discussed in Subsection 11.5.4, with the eventual development of a tool for exemplar-based annotation suggestion. Our analogical transfer approach has several explainability benefits for such a tool, as discussed in Subsection 11.5.5.

11.5.1 Applying the methodology from the ANN framework for analogy solving within and between sentences

Building upon our dataset of semantic APs, and using a very simple methodology, we propose an analogy solving approach that achieves high performance and is able to identify many unsolvable analogical equation, *i.e.*, when the SR is not present in the target sentence. Our approach generalizes well to unseen semantic frames and non-core FEs, as can be seen in Table 11.5.

[10]The model checkpoints we use, `multi-qa-mpnet-base-dot-v1`, `all-mpnet-base-v2` and `all-MiniLM-L6-v2` are provided in the Sentence Transformers library (https://www.sbert.net/docs/pretrained_models.html). `all-MiniLM-L6-v2` is recommended for its execution speed, while `multi-qa-mpnet-base-dot-v1` and `all-mpnet-base-v2` both exhibit high performance for semantic retrieval tasks.

The analogical transfer model displays what can be seen as a tolerance with regards to mistakes in the analogical equation. Firstly, while ideally the source and target sentences activate the same semantic frame, the model maintains high performance for related but distinct semantic frames. Secondly, the formulation of our model requires the first and the third element to instantiate the same SR of the semantic frame, but we show close performance when this rule is not respected.

In future work, we will explore how this tolerance improves performance with regards to mistakes in semantic frame and predicate identification, by actually applying SotA annotation tools to obtain the end-to-end frame semantic parsing pipeline. In such a setting, it wil become possible to identify the impact of mistakes in early steps of the annotation process on the final annotation.

In Subsection 11.3.3, we performed several experiments to determine how the performance is affected by the absence of the A, C elements of the analogical equation $A : B :: C : x$ or the mismatch of the SR they instantiate. Further experiments could be performed to measure to which extent the performance comes from the exact analogical formulation or from the conditioning of the output by adding some A, C related to the semantic frame. For instance, we could experiment with a model trained with A, B, C randomly permuted, or a model with only C as input.

In Subsection 11.3.3, we also experiment with sentences that do not trigger the same semantic frame, and found a correlation between the performance and the SPL of the *Inheritance* relation. For relations other than *Inheritance*, the data generated for this experiment was not well-distributed enough to draw definitive conclusions on the correlation between performance and relatedness. From the nature of frame semantics, our hypothesis is that *Subframe* relations (*Subframe of* and *Has Subframe(s)*) will show similar behavior. An improvement of the process used to generate our data would be to manually align SRs between related semantic frames, instead of considering exclusively the ones with the same label.

11.5.2 Applications to FSRL and the challenge of sentence selection

As can be glimpsed from our experiments, the analogical transfer methodology can outperform current SotA approaches. It is particularly interesting that our method uses an out-of-the-box Ex-QA model, as well as limited amounts of training data. Other SotA approaches, such as the one of Lin, Sun, and Zhang [LSZ21], use span representation to obtain groups of words as candidates for the various FEs, yet our model does not. This is the most likely cause for the a significant amount of *not an SR* mistakes where the predicted boundaries do not correspond to a meaningful FE. In the future, we will update our model in order to better take into the span semantics.

Despite the very encouraging results of our approach under the best conditions, our experiments indicate that the source sentence selection process is an important factor to outperform SotA in practice. As far as FSRL is concerned, our experiments indicate that enforcing a single source sentence for all the FEs of a semantic frame may not be the optimal strategy, in terms of coverage and performance. Instead, a different source sentence may be more appropriate for each FE. Our source selection experiments show that the performance of random selection is far from SotA performance, which is expected. Using high sentence similarity is promising for source selection, however but the number of targets that have similar sources is much too low to achieve the upper bound of the performance we can obtain. Accordingly, in future experiments we will explore additional source selection approaches, for instance by fine-tuning a model specifically for this task.

It would be interesting to explore various approaches including a compromise between the retrieval of a single representative sentence and the use of many sources to maximize coverage. What we envision is the selection of few prototypical and carefully annotated source sentences for each semantic frame, chosen to cover as many FEs as possible and to maximize analogical transfer performance. This idea is similar to the one of prototypes in CBR [Per19]. Additionally, it is likely that using an ensemble of sources for each prediction would improve our model performance. However, this involves significant exploration on the selection of sources and the aggregation of the predictions, and remains a topic of ongoing work.

11.5.3 Dataset of semantic analogical equation within and between sentences

The dataset we built for analogy solving on FN1.7 complements the traditional datasets of semantics analogies between words, that focus on factual and lexical semantics. Indeed, contextual information is usually not considered for APs between words in factual and lexical semantic datasets. Examples of such datasets are the Google dataset [Mik+13], BATS [GDM16], JBATS [Kar+18], some of which are combined in the recent ANALOGICAL [Wij+23] dataset.

In contrast to these datasets, the one we propose can be leveraged for studying analogical transfer for FSRL, exploiting the context in which each SR is situated, *i.e.*, the rest of the phrase. By providing a clear definition of the underlying relation manipulated in the APs, we also provide new insights on the study of semantic APs between and within sentences.

11.5.4 Generalizing the approach to other languages and datasets

As mentioned in Subsection 11.2.1, we used mBert to be able to extend our approach to other languages. Indeed, in the past decade there has been a focus on providing labeled frame semantics resources for languages beyond English with, among other, FrameNet data in French [Dje+16] and Swedish [DBF21]. However, this effort is for the most part limited to languages with many speakers, and frame annotation remains difficult and costly. To tackle this issue, further work will be done to offer a tool for frame semantic parsing leveraging analogical transfer and the multilingual embedding model mBert for languages for which few or no labeled data is available.

Also, in our experiments, we use exclusively FN1.7, but we plan to extend the approach to similar datasets, such as PropBank [Pra+22].

11.5.5 Towards explainable FSRL

Our analogical formulation of FSRL, one could argue, falls into the paradigm of exemplar-based processes. This means it leverages an exemplar, in our case the source sentence and its annotation, to solve the task on the target sentence. Among others, Yasunaga, Chen, et al. [Yas+24] proposed to improve the expandability of modern PLMs by providing to the user the exemplars used for analogical reasoning, in a Chain-of-Thought scenario where steps of reasoning are made explicit. A similar approach could improve the expandability of the results and explainability of FSRL tools, which we will explore in future work.

Chapter 12

Complexity Measure for Analogical Transfer and case (base) competence

Chapter contents

This chapter summarizes the contributions in [Mar+23] and includes further experiments that have not been published at the time of writing.

The Complexity Measure for Analogical Transfer (CoAT) method, introduced by Badra [Bad20], is a Case-Based Reasoning (CBR) method base on the notion of analogical transfer. The method is based on Γ, an indicator of the complexity of a dataset from the point of view of analogical transfer, and allows to answer questions such as: “Is a given similarity measure more suitable than some other (for a task)?” or “How compatible is a solution with a problem, given a case base and similarity measures?”

What we propose in [Mar+23], based on the CoAT indicator Γ, differ from previous work on case base maintenance in three main ways:

1. instead of a focus on local relationships as is frequently done in CBR, Γ is an indicator that considers the entirety of the case base;

2. to determine the usefulness of a case for maintenance purposes, instead of approximating future problems from the case base, we use a set of unseen problems as a reference;

3. the results we obtain question the common assumption that case deletion will result in a loss of performance, and illustrate that compression may actually enhance performance.

12.1 Notions of CBR

This section presents the notations used in this chapter, the key principles of CBR systems, as well as the k-Nearest Neighbors (k-NN) algorithm which is representative of the intuition behind CBR. In the last Subsection 12.1.3, we introduce the notion of case base maintenance, the core topic of [Mar+23] reported in this chapter.

12.1.1 Basic notions and notations

Let $\mathcal{S}$ denote an input space, and $\mathcal{R}$ an output space. An element of $\mathcal{S}$ is called a situation (or problem), and an element of $\mathcal{R}$ is called an outcome (or result or solution). A set $CB = \{(s_1, r_1), \ldots, (s_n, r_n)\}$ of elements in $\mathcal{S} \times \mathcal{R}$ is called a case base. An element $c = (s, r) \in CB$ is called a source case. In addition, the spaces $\mathcal{S}$ and $\mathcal{R}$ are respectively equipped with the similarity measures $\sigma_{\mathcal{S}}$ and $\sigma_{\mathcal{R}}$, that denote a similarity measure on situations and on outcomes.

Let $\mathcal{T} \subset \mathcal{S} \times \mathcal{R}$ be a set of cases called a reference set, and $c_t = (s_t, r_t) \in \mathcal{T}$ be a reference case. We will write $(s_t, r^\star)$ to denote the prediction for the case. The workings of CBR systems is usually decomposed in three sequential tasks [BL23; Gus+08]:

- firstly, the *retrieval* step, where source cases $c = (s, r) \in CB$ are retrieved from the case base;
- secondly, the *mapping* step, where for each situation the similarity $\sigma_{\mathcal{S}}(s, s_t)$ is computed;
- finally, the *transfer* step, where the similarities $\sigma_{\mathcal{R}}(r, r^\star)$ are estimated from the results of the mapping step and the principle of analogical transfer, *i.e.*, that similar situations will have similar outcomes.

After transfer, a plausible outcome is found in order to match the estimated $\sigma_{\mathcal{R}}(r, r^\star)$. To illustrate, we provide an example of a regression task in Example 12.1.

Example 12.1: Example of a case base and a target problem

Let us consider the following example, taken from [Bad20, Table 1]. We want to estimate the rent of appartements based on their number of rooms and the area of the city they are located in. In particular, we want to predict the outcome for an appartement in the downtown, with 2 rooms.

We know the rent of the following apartments, which form our case base:

- $c_1 = (s_1, r_1)$, with $s_1 = (1, \text{midtown})$ and $r_1 = 440$;
- $c_2 = (s_2, r_2)$, with $s_2 = (2, \text{midtown})$ and $r_2 = 600$;
- $c_3 = (s_3, r_3)$, with $s_3 = (1, \text{downtown})$ and $r_3 = 700$;
- $c_4 = (s_4, r_4)$, with $s_4 = (3, \text{downtown})$ and $r_4 = 900$.

Our target case is then $c_t = ((2, \text{downtown}), r_t)$ for which we want to find r_t.

In this setting, the situation space is $\mathcal{S} = \mathbb{N} \times \{\text{midtown}, \text{downtown}\}$, and our outcome space is $\mathcal{R} = \mathbb{R}^+$.

12.1.2 Common methods to CBR

k-Nearest Neighbors (k-NN). The k-NN algorithm is often used in a classification setting. As described for instance in [Aha92], the algorithm retrieves the k cases with the situations most similar to the target situation, as the name indicates. Then it predicts the outcome that has the most support among the retrieved cases, by majority voting. Many variants of k-NN have been proposed, including the one used in [KAÜ19], and a more common veriant weighting the vote by the value $\sigma_{\mathcal{S}}(s, s_t)$, as indicated in [BL23]. For regression problems for instance, the Scikit-Learn implementation offers the option to use a weighted average of the outcomes, the weights being the similarities $\sigma_{\mathcal{S}}(s, s_t)$. The same process was used in Example 12.2.

Approach	Compatibility knowledge	Prediction strategy
Evidence support	A joint similarity measure, that measures how compatible $\sigma_\mathcal{R}$ is with $\sigma_\mathcal{S}$ for a given pair of cases.	Find the case that is most compatible with the retrieved cases.
Continuity constraints	A set of continuity constraints, *i.e.*, rules that state that $\sigma_\mathcal{R}$ should be compatible with $\sigma_\mathcal{S}$ on each pair of cases.	Exclude the outcomes that are not similar enough to the outcomes of the retrieved cases.
Approximate reasoning	A set of rules of the form $(\sigma_\mathcal{S} = \alpha) \rightarrow (\sigma_\mathcal{R} = \beta)$.	Make a majority vote on the outcomes derived from the rules.
Global optimization	A global function, that measures how compatible $\sigma_\mathcal{R}$ is with $\sigma_\mathcal{S}$ on the whole case base.	Optimize the global compatibility measure on the augmented case base.

Table 12.1: Table 1 from [BL23]. Proposed typology of case-based prediction theories.

Example 12.2: Example of k-NN on the setting of Example 12.1

To compare the cases in Example 12.1, we need to define multiple similarity measures. The $\sigma_\mathcal{S}$ defined in the experiments of [Bad20] is the average of two similarities:

$$\begin{aligned}\sigma_\mathcal{S}((nb\_rooms_i, area_i), (nb\_rooms_j, area_j)) &= \frac{1}{2}(\sigma_\mathcal{S}^{nb\_rooms}(nb\_rooms_i, nb\_rooms_j).\\ &\quad + \sigma_\mathcal{S}^{area}(area_i, area_j))\\ \sigma_\mathcal{S}^{nb\_rooms}(nb\_rooms_i, nb\_rooms_j) &= \frac{(6 - |nb\_rooms_i - nb\_rooms_j|)^2}{6^2},\\ \sigma_\mathcal{S}^{area}(area_i, area_j) &= 1 \text{ if } area_i = area_j \text{ else } 0.\end{aligned}$$

Let us now consider the similarity between each case in the case base and $c_t = ((2, \text{downtown}), r_t)$:

- for c_1, we have $\sigma_\mathcal{S}(s_t, (1, \text{midtown})) \approx 0.347$ and $r_1 = 440$;
- for c_2, we have $\sigma_\mathcal{S}(s_t, (2, \text{midtown})) = 0.5$ and $r_2 = 600$;
- for c_3, we have $\sigma_\mathcal{S}(s_t, (1, \text{downtown})) \approx 0.847$ and $r_3 = 700$;
- for c_4, we have $\sigma_\mathcal{S}(s_t, (3, \text{downtown})) \approx 0.847$ and $r_4 = 900$.

With a 1-NN we would retrieve either c_3 or c_4, and predict respectively $r^\star = 700$ or $r^\star = 900$. With a 2-NN we would retrieve c_3 and c_4, and as $\sigma_\mathcal{S}(s_t, s_3) = \sigma_\mathcal{S}(s_t, s_4)$, we would predict $r^\star = 800$. Going further, with a 3-NN we would retrieve c_2, c_3, c_4 for a prediction:

$$r^\star = \frac{600 \times 0.5 + 700 \times 0.847 + 900 \times 0.847}{0.5 + 0.847 + 0.847} \approx 754.$$

Typology of CBR methods. Badra and Lesot [BL23] propose a typology of existing CBR methods based on five criteria, resulting in four families of approaches, summarized in Table 12.1. We do not explore this typology here, but the interested reader can read further in the survey [BL23]. Nevertheless, k-NN is a typical evidence support approach, as a case is considered plausible if there exists a similar case in the case base. The CoAT approach, that we define in Section 12.2, is based on a compatibility indicator Γ optimized over the whole case base, and as such is one of the few global optimization approaches according to [BL23].

12.1.3 Case base maintenance

The principle behind case base maintenance is to revise or organize the content of a case base to improve performance, and has been a longstanding and recurrent area of research in CBR [WL01].

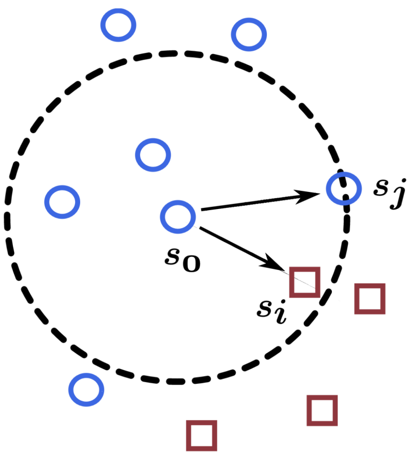


Figure 12.1: Figure 1 from [Bad20]. Inversion of similarity for s_0: s_i is more similar to s_0 than s_j in terms of Euclidean distance, but r_i (red square class) is less similar to r_0 than r_j (both in the blue circle class).

Much of this work has studied case base compression by case deletion [SK95]. This research area is mainly motivated by concerns over the computational cost of retrieval in the case base but also over the maintenance cost of the case base for knowledge engineers. However, blindly deleting cases to reduce the size of a case base may remove knowledge necessary to solve specific situations, resulting in a focus on the notion of competence. To be more precise, for deletion-based strategies, the goal is to identify cases whose removal will harm the competence of a case base the least.

The contribution of a case to competence is commonly estimated within a case base, relying on the assumption that the case base is representative of the distribution of future problems [SM01]. Given the predominance of local strategies in CBR, such as k-NN, the case competence is commonly based on the relationships between cases and their nearest neighbors in the case base. In practice, cases that have high coverage of other cases and that are recoverable from fewer cases, *i.e.*, harder to reconstruct if removed, tend to be kept in the case base [SM01].

12.2 CoAT

In this section, we give the definition of the CoAT method [Bad20; BL22; Bad+22]. CoAT is built around a global indicator Γ that measures the ordinal compatibility between two similarity measures $\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}$. By minimizing Γ, it is possible to determine the most suitable parameters for similarity measures $\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}$, but also to find the most suitable outcome $r_t \in \mathcal{R}$ for a new situation $s_t \in \mathcal{S}$, as described in Subsection 12.2.2. As identified in [Mar+23], it is possible to interpret CoAT in the framework of energy based methods, as described in Subsection 12.2.3.

12.2.1 CoAT indicator Γ

The indicator Γ measures the compatibility of two similarity measures $\sigma_{\mathcal{R}}, \sigma_{\mathcal{S}}$ in a given setting. The compatibility is expressed following an ordinal understanding of the basic analogical principle, according to which similar situations will have similar outcomes, and conversely dissimilar outcomes are associated with dissimilar situations. In practice, Γ quantifies how frequently the orders induced by $\sigma_{\mathcal{R}}$ and $\sigma_{\mathcal{S}}$ agree.

More formally, we consider the following continuity contraint, with three cases $c_0 = (s_0, r_0)$, $c_i = (s_i, r_i)$, and $c_j = (s_j, r_j)$:

$$\sigma_{\mathcal{S}}(s_0, s_i) \geq \sigma_{\mathcal{S}}(s_0, s_j) \implies \sigma_{\mathcal{R}}(r_0, r_i) \geq \sigma_{\mathcal{R}}(r_0, r_j). \quad (C)$$

This constraint expresses that whenever a situation s_0 is more similar to situation s_i than to situation s_j, then its label r_0 must be more similar to r_i than to r_j. Constraint (C) is not satisfied by a triple (c_0, c_i, c_j) if we have an inversion of the consequent:

$$\sigma_{\mathcal{S}}(s_0, s_i) \geq \sigma_{\mathcal{S}}(s_0, s_j) \wedge \sigma_{\mathcal{R}}(r_0, r_i) < \sigma_{\mathcal{R}}(r_0, r_j), \quad (\neg C)$$

in other words, we have an *inversion of similarity* on the triple (c_0, c_i, c_j). An inversion is illustrated in Figure 12.1.

The incompatibility between $\sigma_{\mathcal{R}}$ and $\sigma_{\mathcal{S}}$ for a given case base CB is measured globally as the number of such inversions on the case base CB, by the global indicator denoted $\Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB)$:

$$\begin{aligned}\Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB) = |\{((s_0, r_0), (s_i, r_i), (s_j, r_j)) \in CB^3 \text{ such that} \\ \sigma_{\mathcal{S}}(s_0, s_i) \geq \sigma_{\mathcal{S}}(s_0, s_j) \wedge \sigma_{\mathcal{R}}(r_0, r_i) < \sigma_{\mathcal{R}}(r_0, r_j)\}|. \quad (12.1)\end{aligned}$$

Note that the formulation of Γ can handle tasks beyond classification. For instance, it can handle regression tasks if $\sigma_{\mathcal{R}}$ is a similarity defined on $\mathbb{R}^2$, such as cosine similarity.

Upper bound for the value of Γ. In [Bad+22, Theorem 1], a tight upper bound Γ_{max} for Γ is given, the demonstration given hereafter.

The value of Γ_{max} is derived from the observation that, if we have an inversion for a triple (c_0, c_i, c_j), we will not have an inversion for a triple (c_0, c_j, c_i). Additionally, we never have an inversion for a triple (c_0, c_i, c_i), as in that case $\sigma_{\mathcal{S}}(s_0, s_i) = \sigma_{\mathcal{S}}(s_0, s_i) \wedge \sigma_{\mathcal{R}}(r_0, r_i) = \sigma_{\mathcal{R}}(r_0, r_i)$. The number of inversions for a given c_0 is therefore at most the number of pairs $\{c_i, c_j\} \in CB^2, c_i \neq c_j$ in the dataset, therefore $\frac{|CB|(|CB|-1)}{2}$. As Γ is the sum of such inversions for all possible $c_0 \in CB$, we have:

$$\Gamma \leq \Gamma_{max} = \frac{|CB|^2(|CB|-1)}{2}.$$

It was also shown by Badra, Lesot, et al. [Bad+22] that Γ_{max} is a tight upper bound, as it is reached in some cases.

12.2.2 Applications of CoAT

Γ as a measure of dataset complexity. Dataset complexity measures were introduced as a measure of the difficulty of a classification problem [HB00]. As described in [HB00], Γ was introduced as a measure of how suitable two similarity measures $\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}$ are to perform prediction on a case base CB. In [Bad+22], it was shown that the difficulty of a binary classification task, expressed as the overlap between two classes in synthetic datasets, correlates with the value Γ.

Γ to optimize the configurations of similarity measures. Given that Γ measures the complexity of a prediction task using $\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB$, if $\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB$ has a lower complexity than $\sigma'_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB$, then $\sigma_{\mathcal{S}}$ is more suitable for the task than $\sigma'_{\mathcal{S}}$. In other words, $\sigma_{\mathcal{S}}$ agrees with $\sigma_{\mathcal{R}}$ on CB more than $\sigma'_{\mathcal{S}}$ does.

Using this principle, Γ was used in [Bad20] to compare different configurations of a weighted sum of per-feature similarities. The authors managed to select weights close to the optimum among randomly generated sets of candidate weights for the similarity, by minimizing the value of Γ. A significant correlation was observed between the performance of the k-NN algorithm using a given configuration of the similarity, and the value of Γ for that similarity.

In [Bad+23], we proposed to learn dissimilarities using Γ as an optimization criterion for optimization algorithms.

CoAT for prediction. The principle of CoAT is to use the Γ indicator to predict the outcome r_t of a new situation s_t. The most plausible outcome r_t for a new situation s_t according to a case base CB and some way to compare cases, is the outcome that, when associated with s_t, is the most consistent with CB. Γ indicates the number of inconsistencies between the orders induced by $\sigma_{\mathcal{R}}$ and $\sigma_{\mathcal{S}}$ on a case base CB, therefore, the outcome that minimizes Γ is seen as the most suitable:

$$r_t = \operatorname*{argmin}_{r \in \mathcal{R}} \Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, r)\}). \quad (12.2)$$

In [Bad+22], it was shown the predictive performance of CoAT is correlated with complexity of the dataset measured by Γ. The difficulty of a task, expressed as the overlap between two classes in synthetic datasets, also correlates with both Γ and the accuracy of CoAT.

12.2.3 CoAT as an energy-based model

Energy-based models. Energy-based models are a family of ML models inspired from statistical physics. They are based on the idea that, as stated in a blog post by Huembeli, Arrazola, et al.:

> "An energy-based model is a probabilistic model governed by an energy function that describes the probability of a certain state. [...] The Boltzmann distribution[1] establishes a concrete relationship between energy and probability: low-energy states are the most likely to be observed." [Hue+21]

The article [LeC+06] is as good introduction to the principles described here.

In energy-based models, a parametrized function $E_\theta(x)$, called an energy function, is used to obtain the energy of a given data point x. The conditional version $E_\theta : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ of the energy function affects an energy value $E_\theta(x, y)$ to pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, that associate an input x with an output y. In view of this, $E_\theta(x, y)$ takes low values if the output y is likely to be observed for x, and higher values if x, y are less likely to be observed together.

Energy-based inference. To perform a prediction for a given input x with an energy-based model associated with the energy function E_θ, we want to find the outcome $y^\star \in \mathcal{Y}$ that is the most likely to be observed for x. This correspond to the following optimization problem:

$$y^\star = \operatorname*{argmin}_{y \in \mathcal{Y}} E_\theta(x, y). \tag{12.3}$$

Learning energy-based models. Learning an energy-based model associated with an energy function E_θ corresponds to finding the most suitable set of parameters θ from a dataset $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$. To find such parameters, we want to minimize the energy values associated with the points around the training samples, that corresponds to likely outcomes, and maximize the energy of all other points.

The principle above is the one of contrastive divergence [Hin+06], that consists in optimizing a contrastive loss, such as the Minimum Classification Error loss (MCE) or the hinge loss [LeC+06]. For these two losses, we consider a training sample $(x_t, y_t) \in \mathcal{D}$ and the *most offensive incorrect answer* $(x_t, \overline{y})$ [LeC+06, Equations (8) and (9)], defined as:

$$\overline{y} = \operatorname*{argmin}_{y \in \mathcal{Y},\ y \neq y_t} E_\theta(x_t, y). \tag{12.4}$$

In other words, $\overline{y}$ is the outcome different from y_t that has the smallest energy when associated with x_t. If the outcome space $\mathcal{Y}$ is continuous, $\|y - y_t\|_2 > \epsilon$ is used instead of $y \neq y_t$, for some threshold ϵ.

The MCE is then defined as:

$$\ell_{MCE}(\theta, x_t, y_t, \overline{y}) = E_\theta(x_t, y_t) - E_\theta(x_t, \overline{y}). \tag{12.5}$$

This loss associates a positive loss value to a training sample (x_t, y_t) whenever its energy is higher than the energy of the incorrect sample $(x_t, \overline{y})$, which is larger the larger the difference in energies is. Conversely, if the energy of the training sample (x_t, y_t) is lower than that of $(x_t, \overline{y})$, the model is rewarded in proportion of the difference in energies. The magnitude of ℓ_{MCE} can be interpreted as the prediction confidence of the model.

The hinge loss works similarly, but only associates a loss value to (x_t, y_t) when its energy is not lower by at least a margin λ than the energy of the incorrect $(x_t, \overline{y})$:

$$\begin{aligned} \ell_{\text{hinge}}(\theta, x_t, y_t, \overline{y}) &= \max(0, \lambda + E_\theta(x_t, y_t) - E_\theta(x_t, \overline{y})) \\ &= \max(0, \lambda + \ell_{\text{MCE}}(\theta, x_t, y_t, \overline{y})). \end{aligned} \tag{12.6}$$

To some extent, this can be seen as discarding the "reward" part of ℓ_{MCE}.

[1]The Boltzmann distribution is a probability distribution used to describe the state of a system depending on its energy, in thermodynamics.

The Γ indicator as an energy function. The CoAT method and Γ share a number of similarities with energy-based models. For instance, energy functions and Γ both consider a macroscopic view of the data to offer a measure of how likely an observation is. In fact, the CoAT method can be interpreted in the energy-based model framework as proposed in [Bad+23; Mar+23], with the energy $E_\theta(s_t, r)$ of any new case as the complexity of the case base CB when adding the new case:

$$E_\theta(s_t, r) = \Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, r)\}). \tag{12.7}$$

In this formulation, the parameters of the energy function are $\theta = (\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB)$, the set $\mathcal{X}$ of inputs is the set $\mathcal{S}$ of situations, and the set $\mathcal{Y}$ of outputs is the set $\mathcal{R}$ of outcomes.

The CoAT prediction method also corresponds to the prediction process of energy-based models with this formulation:

$$\begin{aligned} r_t &= \operatorname*{argmin}_{r \in \mathcal{R}} \Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, r)\}) \\ &= \operatorname*{argmin}_{r \in \mathcal{R}} E_\theta(s_t, r). \end{aligned}$$

12.3 CoAT and case (base) competence

Case bases are, together with similarity knowledge, adaptation knowledge, and domain knowledge, one of the main container of knowledge used in CBR [Ric03]. The selection and acquisition of the cases to include in the case base for a task has significant repercussions on the CBR process, in terms of cost but also performance. The question of the selection of cases relevant for the case base can be reformulated as "which cases are the most competent for the task at hand?" The definition of the competence notion can be seen as the formalization of this issue.

In the Γ energy-based model, the energy function $E_\theta(s_t, r)$ is used to compute a (scalar) energy value for each potential outcome r of the new case c_t. The difference between the energy of the predicted outcome and the lowest energy of all other outcomes can be interpreted as a measure of prediction confidence. Our goal is to capture the idea that the competence of a case base should be related to its ability to maximize the prediction confidence. Therefore, a more competent case base should decrease the energy of the correct outcome of a new case and increase the energy of incorrect outcomes. This matches the principle used to learn energy-based models (see Subsection 12.2.3).

In practice, we leverage loss functions of energy-based models for our definition of case base competence, namely MCE and hinge loss. Our measures of competence are defined with regards to a set of reference cases $\mathcal{T}$ distinct from the case base, that allow us to estimate the competence of the model on unseen data instances.

12.3.1 Competence leveraging energy-based models

MCE competence. The first definition of competence we propose, denoted C_{MCE}, relies on the notion of MCE. More precisely, C_{MCE} is the average value of ℓ_{MCE} across the reference set $\mathcal{T}$. As stated in Equations (12.4) and (12.5), ℓ_{MCE} is defined as the difference between *(i)* the energy of the correct outcome and *(ii)* the minimum energy of a reference case if it were assigned a different outcome:

$$\ell_{MCE}(\theta, c_t) = E_\theta(s_t, r_t) - \min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} E_\theta(s_t, \overline{r}) \tag{12.8}$$

$$\begin{aligned} C_{MCE}(\theta, \mathcal{T}) &= -\frac{1}{|\mathcal{T}|} \sum_{c_t \in \mathcal{T}} \ell_{MCE}(\theta, c_t) \\ &= -\frac{1}{|\mathcal{T}|} \sum_{c_t \in \mathcal{T}} (E_\theta(s_t, r_t) - \min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} E_\theta(s_t, \overline{r})). \end{aligned} \tag{12.9}$$

For a correctly predicted instance, $\ell_{MCE}(\theta, c_t)$ is negative. In that case, the magnitude of $\ell_{MCE}(\theta, c_t)$ can be interpreted as the prediction confidence of Γ, as mentioned previously. For an incorrectly predicted instance, $\ell_{MCE}(\theta, c_t)$ is a positive value that corresponds to the extent of the error, *i.e.*, how much the true class is missed.

Overall, lower values of $\ell_{MCE}(\theta, c_t)$ are better, and correspond to higher values of $C_{MCE}(\theta, \mathcal{T})$. In other words, the greater $C_{MCE}(\theta, \mathcal{T})$, the more competent the case base is with regards to the reference set $\mathcal{T}$.

Hinge loss competence. As mentioned in Subsection 12.2.3, the hinge loss modifies the MCE by integrating an additional parameter λ that corresponds to a margin. Similarly, we propose the hinge loss competence C_{hinge} using $\ell_{hinge}(\theta, c_t)$ (see Equation (12.6)) instead of $\ell_{MCE}(\theta, c_t)$:

$$\ell_{hinge}(\theta, c_t) = \max(0, \lambda + \ell_{MCE}(\theta, c_t)) \tag{12.10}$$

$$\begin{aligned} C_{hinge}(\theta, \mathcal{T}) &= -\frac{1}{|\mathcal{T}|} \sum_{c_t \in \mathcal{T}} \ell_{hinge}(\theta, c_t) \\ &= -\frac{1}{|\mathcal{T}|} \sum_{c_t \in \mathcal{T}} \max(0, \lambda + E_\theta(s_t, r_t) - \min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} E_\theta(s_t, \overline{r})). \end{aligned} \tag{12.11}$$

With C_{hinge}, values of $\ell_{MCE}(\theta, c_t)$ lower than $-\lambda$ are not taken into account. Such values correspond to correctly predicted instances (negative ℓ_{MCE} value) with a high prediction confidence (magnitude of ℓ_{MCE} larger than λ). Excluding these values avoids having a few confidently well predicted instances compensate for mispredicted ones, and thus avoids a scaling issue between negative and positive contributions to C_{MCE} as only the negative contributions (to a margin) are accounted for.

Bounds for C_{MCE} and C_{hinge}. Using the tight upper bound Γ_{max} found in [Bad+22] for the value of Γ, we can determine bounds for the values of ℓ_{MCE} and ℓ_{hinge}, which can be transferred to C_{MCE} and C_{hinge}.

First, as in ℓ_{MCE}, ℓ_{hinge} we deal with E_θ and not Γ, we need to account for the fact that $E_\theta(s_t, r) = \Gamma(\sigma_\mathcal{S}, \sigma_\mathcal{R}, CB \cup \{(s_t, r)\})$, and therefore the upper bound $E_{\theta\, max}$ for E_θ is the value of Γ_{max} for a case base of size $|CB| + 1$.

Theorem 12.1: Bounds of the energy function E_θ

For $\theta = (\sigma_\mathcal{S}, \sigma_\mathcal{R}, CB)$, the upper bound $E_{\theta\, max}$ of E_θ is as follows:

$$E_{\theta\, max} = \frac{(|CB| + 1)^2 |CB|}{2}.$$

As the value of Γ is the cardinality of a set, it is a positive integer and so is E_θ, therefore $E_\theta \in [0, E_{\theta\, max}]$.

We can now determine the upper and lower bounds for ℓ_{MCE} and ℓ_{hinge}.

Theorem 12.2: Bounds for ℓ_{MCE}

For $\theta = (\sigma_\mathcal{S}, \sigma_\mathcal{R}, CB)$, $\ell_{MCE} \in [-E_{\theta\, max}, E_{\theta\, max}]$.

Proof By definition:

$$\ell_{MCE}(\theta, c_t) = E_\theta(s_t, r_t) - \min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} E_\theta(s_t, \overline{r})$$

Therefore, if we consider the extreme case where $E_\theta(s_t, r_t)$ is maximal and $\min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} E_\theta(s_t, \overline{r})$ is minimal, we obtain the upper bound:

$$\begin{aligned} \ell_{MCE}(\theta, c_t) &\leq E_\theta(s_t, r_t) \\ &\leq E_{\theta\, max} \quad \text{as } E_\theta \in [0, E_{\theta\, max}]. \end{aligned}$$

Similarly, when $E_\theta(s_t, r_t)$ is minimal and $\min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} E_\theta(s_t, \overline{r})$ is maximal, we obtain the lower bound:

$$\begin{aligned} \ell_{MCE}(\theta, c_t) &\geq - \min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} E_\theta(s_t, \overline{r}) \\ &\geq -E_{\theta\, max} \quad \text{as } E_\theta \in [0, E_{\theta\, max}]. \end{aligned}$$

Theorem 12.3: Bounds for ℓ_{hinge}

For $\theta = (\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB)$, $\ell_{hinge} \in [0, E_{\theta\, max} + \lambda]$ for $\lambda \geq -E_{\theta\, max}$, and $\ell_{hinge} = 0$ for $\lambda < -E_{\theta\, max}$.

Proof By definition, $\ell_{hinge}(\theta, c_t) = \max(0, \lambda + \ell_{MCE})$.

Therefore, the minimum for ℓ_{hinge} is the maximum between 0 and the minimum for $\lambda + \ell_{MCE}$, which is bounded by 0: for $\lambda \leq E_{\theta\, max}$ as $\ell_{MCE} \in [-E_{\theta\, max}, E_{\theta\, max}]$; for $\lambda > E_{\theta\, max}$, the maximum between 0 and the minimum for $\lambda + \ell_{MCE}$, is $\lambda - E_{\theta\, max} > 0$ as $\ell_{MCE} \geq -E_{\theta\, max}$, and 0 is still a lower bound.

Similarly, the maximum for ℓ_{hinge} is the maximum between 0 and the maximum for $\lambda + \ell_{MCE}$, that is $\lambda + E_{\theta\, max}$ for any $\lambda \geq -E_{\theta\, max}$ as $\ell_{MCE} \in [-E_{\theta\, max}, E_{\theta\, max}]$. For $\lambda < -E_{\theta\, max}$, the upper bound becomes 0 as $\lambda + \ell_{MCE} < 0$.

$C_{MCE}(\theta, \mathcal{T})$ and $C_{hinge}(\theta, \mathcal{T})$ are the inverse of the mean of $\ell_{MCE}(\theta, c_t)$ and $\ell_{hinge}(\theta, c_t)$ respectively. The mean over a set is bounded in the same way as the values of the set, and as we consider the inverse we need to inverse the bounds of ℓ_{MCE} and ℓ_{hinge}. Therefore we have $C_{MCE} \in [-E_{\theta\, max}, E_{\theta\, max}]$, as well as $C_{hinge} \in [-\lambda - E_{\theta\, max}, 0]$ for $\lambda \geq -E_{\theta\, max}$ and $C_{hinge} = 0$ otherwise.

12.3.2 Fine-grained notions of competence

We break down the case base competence at the level of the individual cases in the case base and the reference set.

Case competence. The competence of a source case $c = (s, r) \in CB$ can be seen as the contribution of the case to the overall competence of the case base. More explicitly, adding a competent case to the case base would increase the competence of the case base more than adding a less competent case, and conversely, removing a competent case or replacing it with a less competent one would reduce the competence of the case base. Therefore, we propose the competence of a source case c with regards to a reference set $\mathcal{T}$ and parameters $\theta = (\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB)$ as:

$$C(c, \theta, \mathcal{T}) = C(\theta, \mathcal{T}) - C((\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \setminus \{c\}), \mathcal{T}). \tag{12.12}$$

Once again, the larger the value $C(c, \theta, \mathcal{T})$, the more competent the case is.

Case influence and expertise areas. We can further break down the notion of competence by considering the competence with regards to a specific reference case $c_t \in \mathcal{T}$, that we call the influence of the case c with regards to c_t:

$$influence_{\theta}(c, c_t) = \ell(\theta, c_t) - \ell((\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \setminus \{c\}), c_t). \tag{12.13}$$

Interestingly, the competence of a case $c \in CB$ can then be rewritten as:

$$C(c, \theta, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{c_t \in \mathcal{T}} influence_{\theta}(c, c_t). \tag{12.14}$$

This notion of case influence entails the idea of locality: all source cases contribute to the competence of the case base, but each source case may contribute differently on different regions of input space, represented by different instances of the reference set. Case influence can thus be used to identify regions where a source case can improve the performance from those where performance is degraded, that we call the regions of expertise of the source case. Examples of such regions are visualized in Example 12.3.

Example 12.3: Areas of expertize of two extreme cases

Let us consider a binary classification setting using the Half Moon distribution, which is used to produce synthetic examples to test ML algorithms. The principles behind this distribution are further detailed in Subsection 12.4.1.

We visualize a case base as circles and the reference sets as pale triangles in Figure 12.2.

The least and most competent cases are circled the figure, with respectively $C(c, \theta, \mathcal{T}) = -4.680$ and $C(c, \theta, \mathcal{T}) = 4.020$. Interestingly, the least competent case is nested at the core of its class, where other cases can take over if the case is removed. Conversely, the most competent case is the tip of the moon of its class in the case base.

In addition to their position, the areas of expertise (or influence maps) of the two cases are materialized in the background color of Figure 12.2. Negative influence (in red) corresponds to areas where the case harms the performance, while positive influence (in green) corresponds to areas where the case improves the competence. We can see that the least competent case degrades the performance on many reference cases without improving performance on any other case. This least competent case appears harmful to the case base, and is a target for removal as we will see in Subsection 12.3.3. The best performing case is particularly helpful for the competence in the tip of the moon where it is located, which makes sense as the case is close to a key area of the decision boundary. Interestingly, this case also harms the performance for some references of the opposing class.

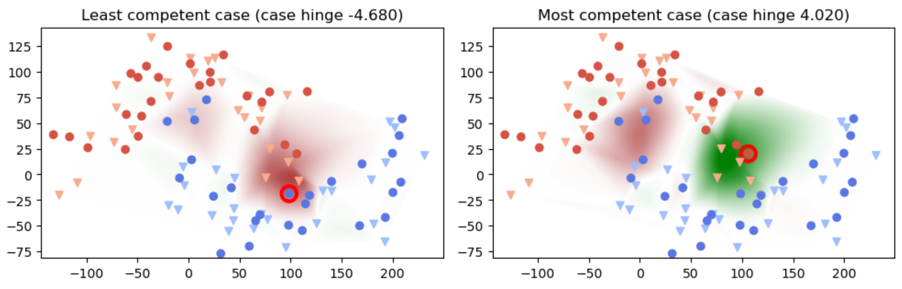


Figure 12.2: Influence map of 2 source cases c_1 and c_2 (circled in red) of the Half Moon dataset (CB=colored disks, $\mathcal{T}$=pale colored triangles): the background color shows, at each position x, y, the value of $influence_\theta(c_1, (x, y))$, estimated using the reference cases. Green corresponds to a positive value of the influence and red to a negative one.

12.3.3 Case base maintenance and case deletion procedure

The competence of a case in the case base can be applied to case base maintenance, either to add new useful cases to the case base or to remove less useful ones, for instance the worst performing case in Example 12.3.

In our work, we focus on case deletion and propose the step by step process illustrated by Algorithm 1. At each iteration, the source case c_{worse} that contributes least to the competence of the case base CB, with regards to the reference set $\mathcal{T}$, is deleted from the case base. In our experiments, we observe the effects of successive deletions, therefore Algorithm 1 is exhaustive. In practice, deletion would repeat only until a stopping criterion is reached, for instance, a desired compression.

Our algorithm is based on the idea that, by fixing $\sigma_\mathcal{S}$ and $\sigma_\mathcal{R}$ in the parameters $\theta = (\sigma_\mathcal{S}, \sigma_\mathcal{R}, CB)$ of CoAT, optimizing E_θ using corresponding loss functions should allow us to learn the right case base CB for the task, *i.e.*, address the case base maintenance issue.

Algorithm 1: Case deletion procedure

```
Input: An initial case base $CB$ and a reference set $\mathcal{T}$
while $|CB| > 0$ do
    $c_{\text{worse}} = \text{argmin}_{c \in CB}\ C(c, CB, \mathcal{T})$;
    $CB = CB \setminus \{c_{worse}\}$;
end
```

12.4 Experiments on synthetic data

In this section, we report the experiments from [Mar+23] on synthetic data. In those experiments, we explore several data distributions and investigate the properties of the case deletion procedure proposed in Subsection 12.3.3. In particular, we compare the two definitions of competence and examine their correlation with the classification performance of CoAT. We perform an analysis of the robustness of the method when placed in different starting configurations of the data distribution. The section is concluded by a qualitative analysis of the results of the experiments.

12.4.1 Experimental setup

Synthetic datasets. We experiment in the binary classification setting, and consider three synthetic two-dimensional datasets. The synthetic datasets are generated from three datasets, namely the Line, Ring, and Half Moon distributions, illustrated in Figures 12.3a, 12.4a and 12.5a and defined as follows.

- The Line data are drawn from a uniform distribution defined on $[0, 2] \times [0, 3]$. They are labeled according to the arbitrary line $f(x) = -x + 2.5$. Noise is then added by randomly switching the label, with a probability of 20%, for cases within a 0.3 distance to the decision boundary.

- The Ring data is made up of two concentric rings of points, each corresponding to one of the classes of the dataset. In our case, the rings have radii 25 and 50. For each class, points are randomly sampled using polar coordinates, drawing the angle from a uniform distribution on $[0, 2\pi]$ and radius from a normal distribution $\mathcal{N}(\mu = r_c, \sigma = 10)$, where $r_c \in \{25, 50\}$ is the radius of the class. The theoretical decision boundary for the Ring data is the circle of radius 32.5.

- The Half Moon dataset is generated with the "make_moons" function from the Scikit-Learn library[2], with a noise of 0.2 and a scale of 100. In practice, the distribution is composed of two halves of a circle of radius $r = 100$, one of which is shifted laterally by the radius. Each half-circle corresponds to a class. Once the points are generated, a two-dimensional Gaussian noise with a scale of $\mathcal{N}(\sigma = 20)$ is added to each data point.

These distribution are among the most frequently used to compare binary classifiers[3], as they cover a range of different decision boundaries.

Remark 12.1

The Ring data we produce is slightly different from the data produced by the "make_rings" function of Scikit-Learn, as the latter generates a ring and then apply two-dimensional Gaussian noise on the resulting points. In our generation process, we apply the noise on the radius when generating the points, and thus maintain the uniform distribution of angular positions of the points in each class.

Similarity measures. The similarities we use are the same for all three synthetic datasets. The source space is $\mathcal{S} = \mathbb{R}^2$, and we use a similarity that is a decreasing function of the Euclidean distance: $\sigma_\mathcal{S}(s_x, s_y) = \exp(-\|s_x - s_y\|_2)$. As we consider binary classification, the outcome space is $\mathcal{R} = \{0, 1\}$, for which we use the class identity similarity: $\sigma_\mathcal{R}(r_x, r_y) = 1$ if $r_x = r_y$ and 0, otherwise.

Due to their geometry, the three data distributions are more or less compatible with $\sigma_\mathcal{S}$. Clever selection of a $\sigma_\mathcal{S}$ dedicated to each situation would increase performance, but maintaining varied levels of compatibility helps us understand the limitation of CoAT in settings where the similarity is not as good as it could be.

[2]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

[3]An example of such comparison can be found in the Scikit-Learn library: https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Protocol. For each of the three data distributions considered, we generate 1000 samples that we split into 20 non-overlapping subsets of 50 cases, balanced in terms of classes. In other words, each subset contains 25 instances of each class, evenly sampled in the distribution. We separate the 20 subsets in 2 groups: 10 serve as initial case bases $CB_1, \ldots, CB_{10}$ and the others as reference sets $\mathcal{T}_1, \ldots, \mathcal{T}_{10}$. The distribution of 1000 samples is shown in Figures 12.3a, 12.4a and 12.5a for the Line, Ring, and Half Moon datasets respectively. The reference cases are shown in a lighter color than the cases from the case bases.

For each dataset, we apply the case deletion procedure (Algorithm 1) on all possible combinations $(CB_i, \mathcal{T}_j)$ of a case base and a reference set, therefore we have 100 runs for each data distribution. We repeat the process using the two proposed definitions of competence, C_{MCE} and C_{hinge}, as the criterion to determine the least competent case for case deletion. After each removal step, we compute for the updated case base the classification performance of CoAT, the C_{MCE}, and the C_{hinge} on all reference cases $\bigcup_{k=1..10} \mathcal{T}_k$. For classification performance, we use using macro F1, *i.e.*, the average of the F1 of each class.

Figures 12.3b, 12.4b and 12.5b compare the evolution of the macro F1 during case deletion using the two proposed competence measures $C_{MCE}(c, \theta, \mathcal{T})$ and $C_{hinge}(c, \theta, \mathcal{T})$. Similarly, in Figures 12.3c, 12.3d, 12.4c, 12.4d, 12.5c and 12.5d we report the evolution of the competence. The shade corresponds to the 95% confidence interval over the 100 combinations of initial case base and references. In Figures 12.3e, 12.4e and 12.5e, each line shows the results for one of the 10 case bases and the shades show the 95% confidence interval over the 10 reference sets. Reciprocally, in Figures 12.3f, 12.4f and 12.5f, each line corresponds to a reference set and the shades correspond to the 95% confidence interval over the 10 initial case bases. The vertical axis for Figures 12.3b, 12.3e, 12.3f, 12.4b, 12.4e, 12.4f, 12.5b, 12.5e and 12.5f is set to the $[50\%, 100\%]$ range of macro F1.

12.4.2 Results

Correlation between the proposed competence notions and the performance of CoAT. In Figures 12.3b, 12.4b and 12.5b, we compare the evolution of the macro F1 when using either C_{MCE} or C_{hinge} as a criterion to determine the least competent case for case deletion. We observe very distinct tendencies when using C_{MCE} rather than C_{hinge} as the compression criterion.

With C_{MCE}, F1 remains at its maximum slightly longer, so a few more cases can be removed. However, with C_{MCE}, F1 remains at its initial value throughout the process and does not reach as high as with C_{hinge}, with even some loss of performance at the beginning of the compression. For instance, on Ring, F1 reaches a value close to 60% for C_{MCE} and 85% for C_{hinge}.

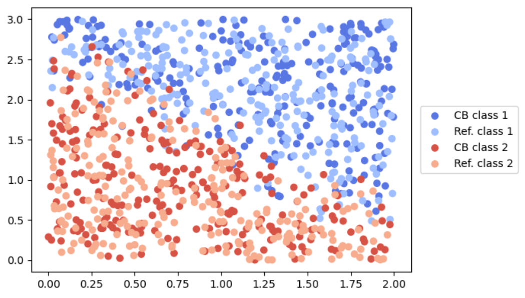
In complementary experiments, we examine the evolution of the case base competence during the deletion process. The evolution of the competence measured by C_{hinge} appears more correlated with the F1 than the value of C_{MCE}. For instance, for the C_{MCE} deletion criterion, in Figure 12.4d we can easily identify the drop in performance that also appears in the first 15 steps in Figure 12.4b, while in Figure 12.4d the value of C_{MCE} is strictly increasing. As mentioned in Subsection 12.3.1, prediction successes and failures are considered at the same time in C_{MCE}, but higher C_{MCE} could be an expression of higher confidence in already well predicted cases, of fewer errors, or of less confident errors. In that regard, C_{hinge} is more suitable as a competence measure: it measures how confident the model is in its errors, and thus higher C_{hinge} corresponds to fewer or less confident errors, which directly translates to higher performance.

In (c) and (d) from Figures 12.3 to 12.5, the value of C_{hinge} spikes up to 0 and the performance drops to 0. This behavior appears only after the last step of deletion, when only one case remains. Indeed, in this specific configuration of the model, it becomes impossible to generate similarity inversions between only two cases, namely, the only case in the case base and the reference.

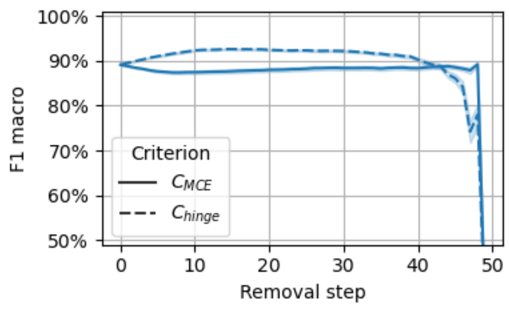
The experiments described hereafter consider only C_{hinge} as the competence measure and as the deletion criterion, as it achieves higher performance, its behavior is more consistent across the data distributions, and is a better fit for the notion of competence.

Case competence and impact on performance. When we consider the evolution of the F1 with the C_{hinge} criterion (see (b), (e), and (f) from Figures 12.3 to 12.5), the same trend of performance can be observed across all initial case bases and the reference sets, for all data distributions. This trend can be separated in three phase:

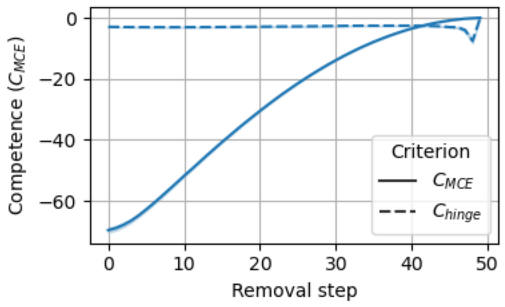
1. a raise;



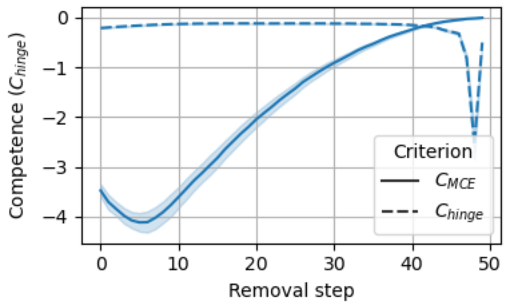
(a) Line distribution



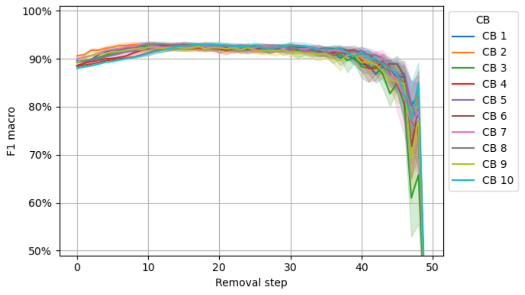
(b) F1 evolution during deletion



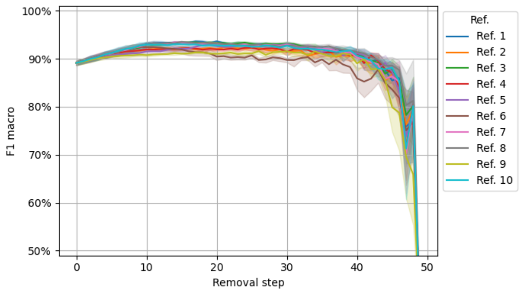
(c) C_{MCE} evolution during deletion



(d) C_{hinge} evolution during deletion

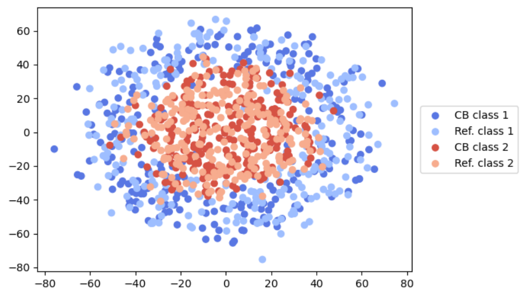


(e) F1 evolution with C_{hinge} for each case base

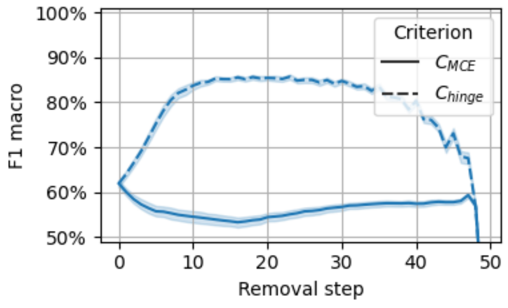


(f) F1 evolution with C_{hinge} for each reference set

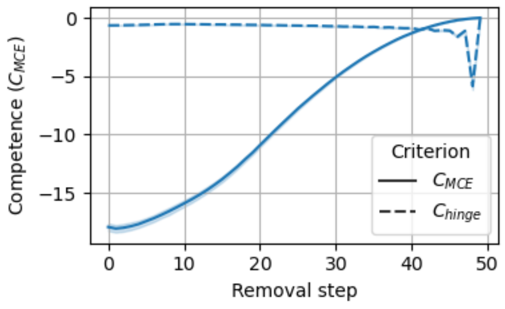
Figure 12.3: Case deletion experiment results for the Line data. The distribution of the 1000 cases used is displayed in (a). In (b), (c), and (d), the performance and competence when using C_{MCE} or C_{hinge} as a criterion are compared. The performance with C_{hinge} is also detailed when grouping by case base initialization (c) and reference set (d). The three plots (b), (e), and (f) display the evolution of the macro F1 (measured on all 500 reference cases) during the case deletion procedure, which consists in 49 steps.



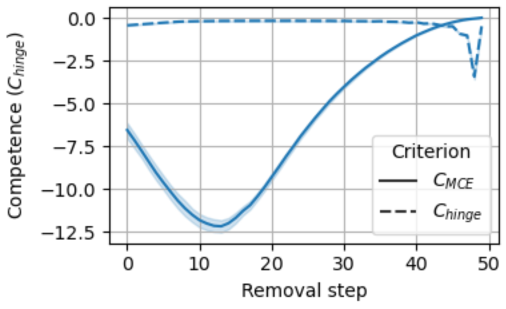
(a) Ring distribution



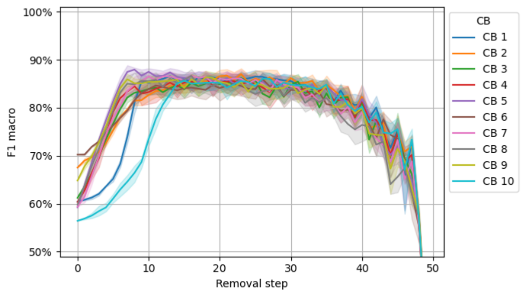
(b) F1 evolution during deletion



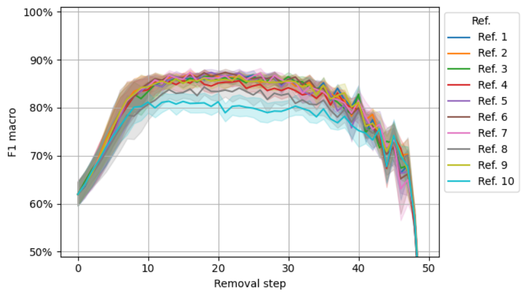
(c) C_{MCE} evolution during deletion



(d) C_{hinge} evolution during deletion

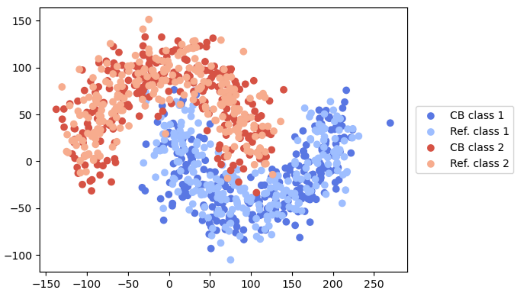


(e) F1 evolution with C_{hinge} for each case base

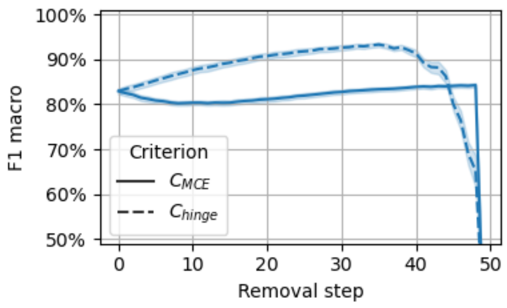


(f) F1 evolution with C_{hinge} for each reference set

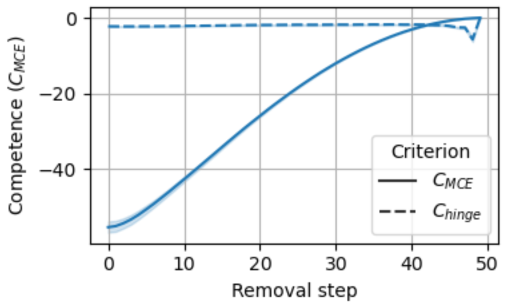
Figure 12.4: Case deletion experiment results for the Ring data. The distribution of the 1000 cases used is displayed in (a). In (b), (c), and (d), the performance and competence when using C_{MCE} or C_{hinge} as a criterion are compared. The performance with C_{hinge} is also detailed when grouping by case base initialization (c) and reference set (d). The three plots (b), (e), and (f) display the evolution of the macro F1 (measured on all 500 reference cases) during the case deletion procedure, which consists in 49 steps.



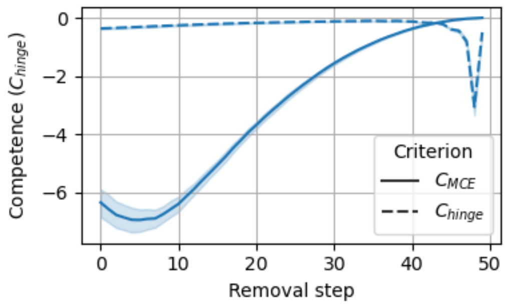
(a) Half Moon distribution



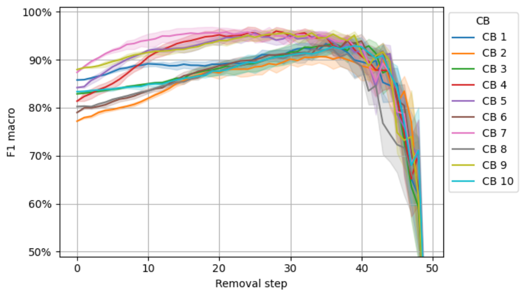
(b) F1 evolution during deletion



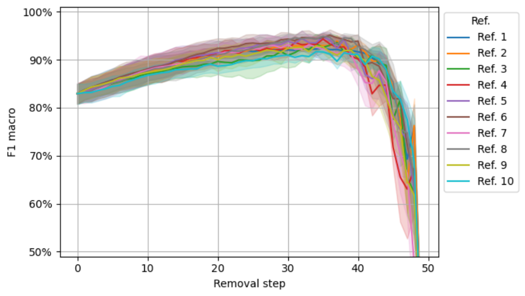
(c) C_{MCE} evolution during deletion



(d) C_{hinge} evolution during deletion



(e) F1 evolution with C_{hinge} for each case base



(f) F1 evolution with C_{hinge} for each reference set

Figure 12.5: Case deletion experiment results for the Half Moon data. The distribution of the 1000 cases used is displayed in (a). In (b), (c), and (d), the performance and competence when using C_{MCE} or C_{hinge} as a criterion are compared. The performance with C_{hinge} is also detailed when grouping by case base initialization (c) and reference set (d). The three plots (b), (e), and (f) display the evolution of the macro F1 (measured on all 500 reference cases) during the case deletion procedure, which consists in 49 steps.

2. a plateau;
3. a faster and faster decrease.

What differs between the different datasets and starting configurations is the amplitude and duration of each step, and in extreme cases one of the first two phases can disappear. For instance, in the Line distribution the increase is small ($\leq 3\%$) and of a short duration (10 steps), while in Half Moon the increase is more significant ($\geq 10\%$) and spans most of the process (around 35 steps in most cases), which results in the disappearance of the plateau.

This trend can be put in parallel with the evolution of the competence, as during phase 1 removing cases improves the performance and C_{hinge} competence (see Figures 12.3d, 12.4d and 12.5d), meaning that the removed cases had a negative contribution to the competence and were "polluting" the case base. In phase 2, the cases removed do not particularly harm nor benefit the performance nor competence of the case base, as such they can be considered redundant with the other cases of the case base. The cases that remain after phase 2 are the most competent and useful ones, and in phase 3 they are removed by order of increasing competence, leading to sharper and sharper drops in performance.

Furthermore, the general trend provides empirical guarantees that the maximum performance is reached just before the first significant decrease in performance, meaning we can stop the process as soon as we detect such a decrease.

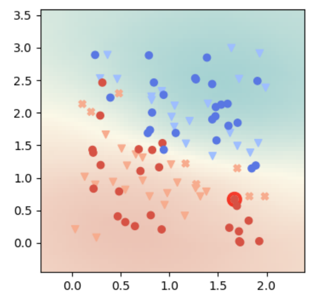
This behavior is similar to the footprint deletion procedure from Smyth and Keane [SK95]: what corresponds to the auxiliary, spanning, and support cases are removed in phase 2, and pivotal cases are removed in phase 3 of our deletion procedure. Auxiliary and spanning cases are cases that are not affecting competence, as other cases can fulfill their role. The main difference between the two is that auxiliary cases are located in areas of the space for which the case base already contain other cases while spanning are located in "holes" in the distribution of the case base. Support cases are a special type of spanning cases. Finally, pivotal cases are cases that are necessary to solve specific problems. The case deletion procedure from [SK95] can bee seen as less powerful and flexible than the one we propose, as the latter can handle cases harmful for the case bases by removing them in priority during phase 1. As a result, our approach is free from the assumption that the initial case base is a good fit for the distribution of the data, which is a frequent assumption in case base maintenance [SK95].

Robustness of the approach. Figures 12.3e, 12.4e and 12.5e show that the initial cases in the case base impact the initial performance and time needed to converge to the general trend of performance. In extreme cases of poor initial performance, the convergence might be delayed until after performance starts to decrease. This can be seen in Figure 12.5e for the lower of the two groups of case bases that appear after step 20, as the lower group does not converge fast enough to reach the maximal performance of the other group of case bases.

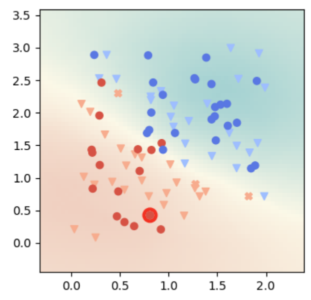
By analyzing the distribution of each set of initial cases we observe that the case base has difficulties to reach the best performance when not enough cases are present in a particular area of the data distribution, *i.e.*, when there are "holes" in the case base in important areas of the situation space. We were able to confirm this effect by manually removing cases in parts of the distribution. Conversely, if we manually make one class over-represented, the performance is not damaged as much, as the cases in the over-represented class are redundant and are removed in the plateau 2.

From these results, an uneven distribution of the cases in the case base at the start of the deletion process harm the best performance only when the initial performance is too poor, leading to converging too slowly to reach the best performance, or when there are no cases in an important area of the boundary.

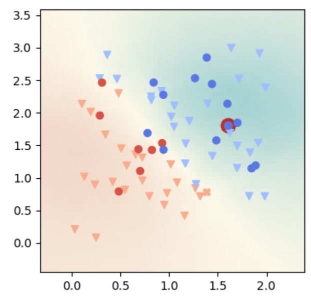
The reference cases, used to measure the competence, also have a significant impact on the best performance reached. Even if there is no major gap between the distribution of references and the true distribution of the data, the maximal performance reached can differ significantly depending in the distribution of the reference set as can be observed with the cyan reference set in Figure 12.4e for the Ring data. In comparison, for the Line and Half Moon distributions, in Figures 12.3e and 12.4e, the differences between the reference sets is less contrasted. However, the effect of the references becomes striking when we manually create holes in the distribution of references. In that setting, the case base becomes biased towards the incorrect distribution of the references, in particular if the distribution of the case base is uneven.



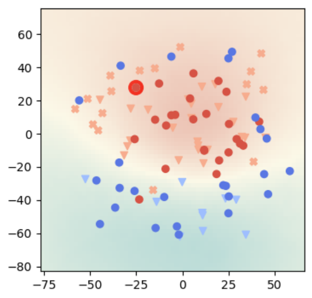
(a) Initial case base (Line). $C_{hinge} = -54.8616$, macro $F_1 = 84.0\%$.



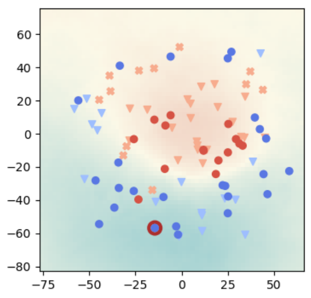
(b) After 10 deletions (Line). $C_{hinge} = -9.5806$, macro $F_1 = 94.0\%$.



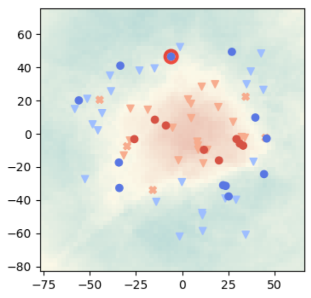
(c) After 30 deletions (Line). $C_{hinge} = -0.0202$, macro $F_1 = 98.0\%$.



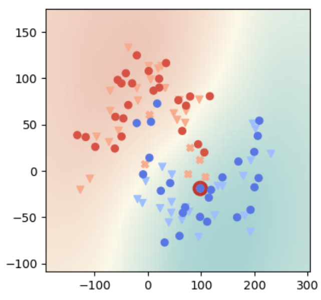
(d) Initial case base (Ring). $C_{hinge} = -105.2632$, macro $F_1 = 65.3\%$.



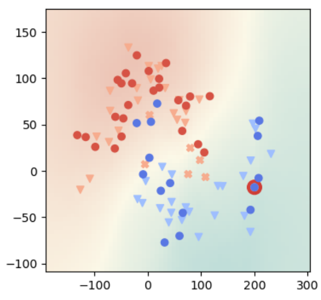
(e) After 10 deletions (Ring). $C_{hinge} = -12.4024$, macro $F_1 = 75.8\%$.



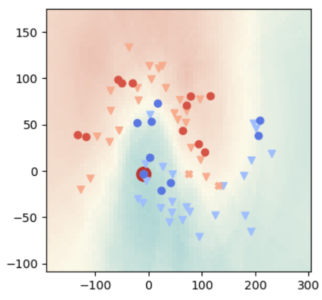
(f) After 30 deletions (Ring). $C_{hinge} = -0.8610$, macro $F_1 = 89.9\%$.



(g) Initial case base (Half Moon). $C_{hinge} = -60.8212$, macro $F_1 = 88.0\%$.



(h) After 10 deletions (Half Moon). $C_{hinge} = -21.5812$, macro $F_1 = 88.0\%$.



(i) After 30 deletions (Half Moon). $C_{hinge} = -0.1804$, macro $F_1 = 96.0\%$.

Figure 12.6: Three steps of the case deletion procedure, for the Line, Ring, and Half Moon data. The background color corresponds to the decision of the model in each area of the situation space, and reference cases are represented as either triangles if they are correctly predicted, or crosses if they are wrongly predicted by CoAT. The case circled in red is the one that will be removed from the case base during the following deletion step.

Qualitative analysis. Figure 12.6 displays 3 steps of the case deletion procedure (initial step, and after 10 and 30 deletions) for one combination of a case base and a reference set, for each of the three data distribution. In each figure, the red and blue dots represent the remaining source cases, and the crosses and the triangles represent the references (triangles and crosses respectively correspond to correct and incorrect predictions by CoAT). The least competent source case c_{worse} that will be deleted is circled in red. The colored map in the figure represents the predictions of CoAT for new cases across the situation space, with the color matching the predicted class and the saturation corresponding to the confidence, *i.e.*, the energy difference between the two classes. In that manner, it is possible to identify the decision boundary of the compressed case base.

At the end of the compression process, the decision frontier for CoAT meets the theoretical classification boundary of the distribution, even for Half Moon which has a relatively complex boundary, and for Ring which as a relatively poor initial performance. The decision frontier induced by the compressed case base is thereby able to closely approximate the ideal classification boundary.

12.5 Conclusion, discussion, and perspectives

In [Mar+23], we introduced an energy-based formulation for CoAT and ways to measure the competence of a case base for ML tasks, such as case prediction and classification. This competence approach differs from prior approaches proposed in the literature as it relies on the optimization of a global compatibility indicator between two similarity measures, one on the situation space and the other on the outcome space. In addition to the results presented in [Mar+23], we provide bounds for the value of the energy function and the competence measures, although these bounds might not be tight.

In Appendix B.1, we present preliminary experiments on real-world datasets, including heterogeneous situation spaces containing both numerical and categorical features. In those experiments, we observe that the performance of CoAT follows the same three phases as on synthetic datasets.

Fast competence computations using Measure of the complexity of a dataset for Analogical Transfer using slices of Boolean Cubes (MeATCube). We detail in Appendix B.2 the MeATCube implementation, that speeds ups computations by several orders of magnitude, significantly improving our ability to experiment with CoAT. Indeed, without optimization, computing a prediction with CoAT has a time complexity in $O(|CB|^3|\mathcal{R}|)$, and computing competence is of the order of $O(|CB|^4|\mathcal{T}||\mathcal{R}|)$. This computational cost can be prohibitive, in particular if any of $CB, \mathcal{T}, \mathcal{R}$ is large. In MeATCube, we combine the optimization proposed by Badra, Lesot, et al. [Bad+22], that divides the time complexity by $|CB|$, with a reformulation of CoAT using high dimensional tensors in PyTorch. By leveraging several optimizations and the parallelization capabilities of PyTorch, we obtain a speedup of two orders of magnitudes compared to the optimized implementation from [Bad+22].

A close relation between competence and the performance of CoAT. We show empirically that this notion of competence is tightly related to performance for a case-based binary classification task, in the sense that the competence of a source case is positively correlated to its ability to reduce the energy of correct outcomes and to increase the energy of incorrect outcomes. We analyze both quantitatively and qualitatively the behavior of a compression algorithm based on the proposed competence measures, on different datasets with substantially different distributions and taking into account different classification frontiers and compression criteria. Moreover, we analyze the robustness of the approach with respect to different reference and initial cases.

A very encouraging observation from the experiments in Subsection 12.4.2 is that, when considering C_{hinge} as the competence measure and criterion, the competence starts decreasing at a similar time as the performance, although it might be difficult to identify due to the scale of the plots. Further experiments and theoretical guaranties would help in devising a stopping criterion for the case base compression algorithm based on this observation.

Comparison with other CBR approaches. These results suggest the strong potential of this energy-based framework for guiding case base maintenance, providing an alternative to existing methods. One of the main differences is that it employs a global approach by considering the competence of a case base as a whole, rather than a local approach as it is often the case in the literature (where only nearest neighbors are considered). The empirical and thorough comparison between the former and the latter will constitute one of the topics to be investigated in a future contribution.

From our preliminary experiments on real-world datasets, presented in Appendix B.1, it appears that the performance of other CBR approaches, such as k-NN, do not follow the performance of CoAT during the case deletion process. However, due to some limitations in our experimental setting, these observations need confirmation.

Extension to multi-class classification and regression. The compression process using C_{hinge} is able to reduce the number of cases in the case base to 40% (Ring) or even 20% (Line and Half Moon) of its initial size, while strictly improving performance. While our current experiments only cover binary classification, our approach is designed to handle any kind of nominal data in the outcome space. We perform preliminary experiments on multi-class classification and real-world data, but further work is still required in that direction. Additionally, the formulation of the competence we use is based on the notions from energy-based models, which are also defined for continuous output spaces. An interesting, although potentially challenging extension of our work would be to adapt the approach for regression tasks.

Selection of the initial case base and reference set. The robustness experiments show that the initial case base is not a major factor in the peak performance, as long as there are enough cases in the important regions of the situation space. However, it is important to have a proper set of reference cases, as the distribution of the references is closely matched by the compressed case base. If the reference cases are not representative of the true distribution of the data, then the compressed case base is not guaranteed to match the true distribution. To summarize, it is useful to focus on the quality of the reference (*i.e.*, how representative of the actual distribution they are) and on having sufficient initial cases for the case base, even if their quality is rather poor, as long as they cover enough of the distribution for the intended purpose of the model.

Proposed competence and CBR methods beyond CoAT. We obtain empirical evidence of the benefits of C_{hinge} over C_{MCE}, and the performance of the case base measured by C_{hinge} correlates to CoAT's prediction performance. The question of whether this measure of competence is compatible with other CBR processes than CoAT remains open, in particular since CoAT and our competence measure are based on the same energy function. We perform preliminary experiments comparing different prediction algorithms through the compression with C_{hinge} as a criterion, but the results are not extensive enough to draw definitive conclusions, in particular since our implementation of approaches beyond CoAT is lacking. Additionally, our competence might be based on the same energy function as CoAT, but we could try to define energy functions suitable for different CBR algorithms and see if the results of energy-based competence generalize.

Theoretical guarantees for CoAT. As a parametrized prediction algorithm, CoAT can be used to optimize similarity measure [Bad20] and the case base, for instance through compression. However, many of the properties of CoAT have not been explored, and we might be able to offer guarantees on the optimization of the parameters θ based on E_θ.

For instance, the ordering of cases based on their competence may change after a case is removed, as the competence measured for every other case before deletion involves the deleted case. This might have an effect on the compression process, but our energy-based approach to competence may offer theoretical guaranties or bounds on those changes. If the competence of a case remains stable when removing another case, we can speed up convergence by removing cases by batches.

Chapter 13

Conclusion on the work presented in this document

In this thesis, we presented our contributions on the ANN framework in morphology [Als+21a; Als+21b; Als+21c; Cha+22; Mar+22a; MC24; Mar+22b] in Part II, while Part III regroups our contributions to TSV [Zer+22], frame semantics, and CBR [Bad+23; Mar+23]. This chapter briefly summarizes these contributions.

The ANN framework on word morphology. In Part II, we proposed the ANN framework to tackle analogy detection and analogy solving on APs where the elements are words and the underlying relations are of a morphological nature.

The analogy manipulation models, called ANNc for analogy detection and ANNr for analogy solving, outperform the symbolic baselines (Nlg, Alea, and Kolmo) and the vector-based approaches (parallelogram rule, 3CosMul) in general. In particular, ANNc and ANNr are able to handle quadruples that do not exactly fit the postulates of APs, and the two embedding models we propose (CNN-emb and AE-emb) are able to deal with morphological features that are context-dependent. This allows our models to tackle APs and morphological transformations that were hard to handle for the symbolic baselines.

We developed Siganalogies as a dataset of morphological AP on more than 80 languages, which enabled interesting comparative studies and extensive testing of the ANN framework. We used a data augmentation that translates the postulates of APs to train DL models using data from the Siganalogies dataset. As such, it can be said that the notion of AP modeled by the models is the intersection of the morphological relations present in the data and the postulates of APs. In particular, we were able to show that altering the data augmentation to account for different sets of postulates resulted in models reflecting the new sets of postulates. In particular, we were able to obtain models corresponding to different sets of postulates by adapting the data augmentation. This led us to ponder on possible combinations of postulates to adapt to different use cases, detailed in Appendix D.

The Siganalogies dataset and its many languages allowed us to explore how the CNN+ANNc model transfer between different data domains (languages) that share latent mechanisms (morphological transformations and the postulates of APs). Among other results, by training on multiple languages at once, we were able to improve the stability and transferability between languages of the model, without increasing the number of training examples. We also correlated the performance of CNN+ANNc with the hierarchy obtained from languages families.

To complement our work, we performed several ablation studies and other experiments to improve the data augmentation and training procedure of our models.

The ANN framework on sentences and semantics. Following the success of the approach on morphology, we applied the data augmentation procedure and ANNc on embeddings produced by PLMs, to tackle to two applications on semantics involving sentences.

Firstly, in Chapter 10, ANNc was able to obtain SotA in TSV. This work allowed us to tackle several challenges related to reformulating a classification task (TSV in this case) as an analogy detection problem, and to the manipulation of heterogeneous APs where the elements are of different nature.

Secondly, in Chapter 11, we used a light-weight model (a pair of perceptrons on top of mBert embeddings) to transfer FE labels from one sentence to another using analogy solving, in the context of frame semantics. Once again, we had to reformulate the problem we tackled, namely, the FSRL task, as analogical equation. Our simple model was able to outperform the SotA, that uses a much more complex model, without having any explicit knowledge of the manipulated labels. However, this was only possible under a the condition that the most suitable source sentence was used for the analogical transfer, a challenge that remains open in our work.

Analogical tranfser applied to CBR. Finally, as described in Chapter 12, we extended the CoAT approach to case (base) competence and case base compression. More specifically, we defined measures of the competence (which can be seen as the relevance or helpfulness to the task) with different levels of granularity (competence of a case base, of a case, and contribution of a case to a specific reference case).

Using these measures of competence, we defined a simple case base compression procedure, that iteratively removes the least competent case of a case base, with regards to some reference cases. By removing cases that have a negative impact on the overall competence of the model according to the measures we defined, we achieved a combination of a reduction of the size of the case base with impressive improvements in performance for CoAT.

We also performed ablation studies on synthetic data, that allowed us to demonstrate the versatility of the approach (in particular its ability to handle complex decision boundaries) and determine the specificities of the system's behavior on edge cases.

Appendix

Appendix A

Appendix to Part II

Language	k	CNN+ANNr	CNN+3CosMul	CNN+ANNc
		Sig16		
	1	**97.60 ± 0.23** **	85.58 ± 7.37 *	76.77 ± 9.57
Georgian	3	**99.54 ± 0.23** **	93.34 ± 3.82 *	92.07 ± 4.92
32233 words	5	**99.73 ± 0.14** **	94.92 ± 2.63	95.14 ± 3.16 *
	10	**99.84 ± 0.07** **	96.19 ± 1.55	97.31 ± 1.68 *
	1	**89.06 ± 1.71** **	74.89 ± 5.27 *	72.95 ± 8.02
Hungarian	3	**97.49 ± 0.42** **	89.04 ± 3.87 *	87.96 ± 5.84
21071 words	5	**98.62 ± 0.29** **	92.40 ± 3.29 *	91.75 ± 4.70
	10	**99.21 ± 0.16** **	94.97 ± 2.49	95.01 ± 3.36 *
	1	**84.75 ± 2.04** **	52.42 ± 4.33 *	44.43 ± 13.95
Turkish	3	**98.20 ± 0.48** **	68.29 ± 4.04 *	63.64 ± 12.89
17225 words	5	**99.28 ± 0.20** **	73.95 ± 3.43 *	70.90 ± 11.30
	10	**99.67 ± 0.09** **	79.96 ± 2.71 *	79.22 ± 8.81
		Sig19		
	1	**93.37 ± 0.97** **	58.01 ± 8.66	80.11 ± 5.10 *
Adyghe	3	**99.63 ± 0.13** **	74.78 ± 7.80	95.93 ± 1.84 *
11155 words	5	**99.85 ± 0.07** **	79.44 ± 7.22	98.08 ± 0.97 *
	10	**99.95 ± 0.03** **	84.02 ± 6.46	99.30 ± 0.41 *
	1	**72.08 ± 3.49** **	21.67 ± 5.44	40.73 ± 9.93 *
Arabic	3	**91.33 ± 1.94** **	38.61 ± 7.44	65.41 ± 11.60 *
12942 words	5	**95.19 ± 1.14** **	47.04 ± 8.23	74.75 ± 11.07 *
	10	**97.82 ± 0.53** **	58.14 ± 8.82	84.30 ± 9.44 *
	1	57.63 ± 5.48 *	37.76 ± 11.12	**65.38 ± 6.01** **
Bashkir	3	78.50 ± 3.00 *	55.72 ± 12.65	**83.85 ± 3.96** **
9231 words	5	87.12 ± 2.04 *	64.30 ± 11.48	**89.61 ± 2.40** **
	10	**98.50 ± 0.50** **	74.58 ± 8.39	94.27 ± 0.86 *

Table A.1: Appendix Tables A6 to A9 from [MC24]. Hit rate at k (in %, mean ± std.) at the word level and number of candidate words for the retrieval models. **: highest average performance; *: second highest average performance.

Language	k	CNN+ANNr	CNN+3CosMul	CNN+ANNc
		Sig19		
English 16245 words	1	**92.29 ± 0.91** **	66.83 ± 15.67 *	65.51 ± 6.25
	3	**98.66 ± 0.29** **	76.34 ± 14.94	80.50 ± 4.66 *
	5	**99.01 ± 0.22** **	79.63 ± 14.15	85.14 ± 3.89 *
	10	**99.31 ± 0.16** **	83.26 ± 13.18	89.99 ± 2.95 *
French 15220 words	1	**93.15 ± 0.96** **	80.37 ± 7.97 *	62.87 ± 17.65
	3	**98.33 ± 0.30** **	91.60 ± 4.64 *	82.94 ± 13.10
	5	**98.86 ± 0.18** **	93.91 ± 3.57 *	88.61 ± 9.81
	10	**99.25 ± 0.13** **	95.94 ± 2.50 *	93.62 ± 6.07
Hebrew 8957 words	1	**66.52 ± 2.59** **	15.47 ± 10.12	36.10 ± 4.28 *
	3	**88.06 ± 2.09** **	30.20 ± 15.80	59.69 ± 4.93 *
	5	**93.13 ± 1.43** **	40.07 ± 17.04	69.63 ± 4.61 *
	10	**96.96 ± 0.78** **	54.58 ± 15.98	80.83 ± 3.77 *
Portuguese 12921 words	1	**93.12 ± 1.10** **	58.52 ± 18.48	70.53 ± 9.88 *
	3	**99.06 ± 0.21** **	79.18 ± 17.98	91.58 ± 6.02 *
	5	**99.50 ± 0.17** **	85.79 ± 15.20	96.26 ± 3.61 *
	10	**99.74 ± 0.10** **	91.98 ± 10.44	98.55 ± 1.50 *
Sanskrit 8473 words	1	**64.18 ± 2.62** **	33.20 ± 9.34	42.59 ± 4.88 *
	3	**89.95 ± 1.26** **	53.90 ± 8.92	66.03 ± 4.88 *
	5	**95.90 ± 0.48** **	63.92 ± 7.26	76.12 ± 4.11 *
	10	**98.78 ± 0.24** **	75.24 ± 4.99	86.48 ± 3.03 *
Slovak 7442 words	1	**56.23 ± 4.57** **	49.43 ± 3.13 *	39.58 ± 3.37
	3	**87.86 ± 1.56** **	67.81 ± 4.08 *	64.46 ± 4.00
	5	**96.50 ± 0.46** **	74.26 ± 4.15	74.42 ± 3.83 *
	10	**98.77 ± 0.31** **	80.71 ± 3.73	84.65 ± 3.10 *
Slovene 10189 words	1	**71.99 ± 2.24** **	57.79 ± 8.59 *	51.88 ± 6.69
	3	**92.28 ± 0.72** **	78.08 ± 8.26 *	75.92 ± 6.94
	5	**97.48 ± 0.28** **	84.38 ± 7.26	84.41 ± 5.89 *
	10	**99.16 ± 0.19** **	89.77 ± 5.58	91.35 ± 4.13 *
Swahili 6419 words	1	**68.56 ± 6.09** **	44.84 ± 7.77 *	44.46 ± 3.85
	3	**84.72 ± 4.70** **	60.57 ± 7.08 *	57.34 ± 5.10
	5	**90.08 ± 3.70** **	66.97 ± 7.03 *	63.30 ± 5.61
	10	**95.42 ± 2.17** **	75.30 ± 6.64 *	71.77 ± 5.76
Welsh 8820 words	1	**63.80 ± 3.13** **	47.30 ± 4.67	47.58 ± 5.72 *
	3	**88.29 ± 1.92** **	71.66 ± 5.04	76.67 ± 6.09 *
	5	**93.88 ± 1.64** **	79.10 ± 4.40	85.76 ± 4.72 *
	10	**97.56 ± 0.84** **	85.97 ± 3.66	93.26 ± 2.71 *
Zulu 9616 words	1	**76.59 ± 2.65** **	58.53 ± 4.42 *	42.56 ± 6.55
	3	**89.66 ± 1.66** **	77.92 ± 4.10 *	64.51 ± 7.17
	5	**92.83 ± 1.21** **	84.28 ± 3.62 *	73.67 ± 6.59
	10	**95.96 ± 0.64** **	90.70 ± 2.80 *	84.30 ± 5.04

Table A.2: Appendix Tables A6 to A9 from [MC24], second part of Table A.1. Hit rate at k (in %, mean ± std.) at the word level for the retrieval models. **: highest average performance; *: second highest average performance.

Appendix B

Appendix to Chapter 12

Chapter contents

B.1 Preliminary experiments on the performance of CoAT on real-world data

In Section 12.4, we present extensive experiments on the behavior of CoAT and the case deletion procedure on synthetic binary classification data in $\mathbb{R}^2$. However, our approach is designed to handle any kind of nominal data in the outcome space, and any kind of data in the situation space, as long as a similarity measure is defined on each of these spaces. To confirm whether we observe the same tendencies on nominal and heterogenous data with a less artificial distribution, we apply the case deletion procedure based on C_{hinge} on well studied real-world datasets.

Note however that these experiments are preliminary, and while they provide useful indications on the performance of CoAT and the case deletion procedure, they are not enough to draw definitive conclusions. We discuss in more details the main limitations in Appendix B.1.3, and additional extensions are mentioned in Section 12.5.

B.1.1 Experimental setup

Datasets. We consider a total of 17 dataset in our preliminary experiments, detailed in Table B.1. These datasets cover binary classification, but also classification with up to 7 classes. The number of instances ranges from 24 to 768, and the nature of the data is varied, with datasets containing only nominal features, only numeric data, or an heterogenous mix of the two. The number of features is also varied, with 3 to 56 features, and datasets with many and few features present for both numeric and nominal data.

We randomly split the data into 60% for the training set, 20% for the development set, and 20% for the test set.

Similarity measures. For the outcome space, we use the class equality similarity we used in Section 12.4 on our synthetic datasets.

For the situation space, we use an aggregation of different similarities for each feature. For symbolic features, we use the class equality similarity, and for numeric features, we use a function

Dataset	Source	Instances	Num. of features Nominal	Num. of features Numeric	Num. of output classes
Balance	UCI	625	0	4	3
Breast Cancer Diagnostic	UCI	569	0	30	2
Breast Cancer Pronostic	UCI	194	0	33	2
Credit Approval	UCI	653	9	6	2
Dermatology	UCI	358	33	1	6
Glass Identification	UCI	214	0	9	6
Haberman's Survival	UCI	306	0	3	2
Heart Disease Cleveland	UCI	297	6	7	5
Hepatitis	UCI	80	13	6	2
Ionosphere	UCI	351	0	34	2
Iris	UCI	150	0	4	3
Lenses	UCI	24	4	0	3
Liver Disorders	UCI	345	0	6	2
Lung Cancer	UCI	27	56	0	3
Pima Indians Diabetes	Kaggle	768	0	8	2
Post-Operative Patient	UCI	87	7	1	4
Teaching Assistant Evaluation	Kaggle	151	4	1	3
Wine	UCI	178	1	12	3
Zoo	UCI	101	16	0	7

Table B.1: Datasets considered for our real-world experiments. We indicate whether the data can be found in the UCI machine learning repository (UCI), or in the Kaggle repository. We also indicate the number of instance after removing instances with missing values, as well as the number of features

of the normalized absolute distance as follows, with $\mathcal{X}^\star$ the observed values for the attribute:

$$\sigma(x,y) = 1 - \frac{|x-y|}{\max(\mathcal{X}^\star) - \min(\mathcal{X}^\star)}. \tag{B.1}$$

To aggregate the similarities for each features, we use a weighted similarity, as done usually for CBR [Bad20; KAÜ19]. The weights were computed on the whole dataset, using the method from Karabulut, Arslan, and Ünver [KAÜ19], without distinguishing between the training, development and test sets. The algorithm to compute the weights is as follows:

- the set $C_i(a)$ of values for attribute a belonging to class i is computed using:

$$C_i(a) = \{X[k][a] : X[k] \in X \text{ and } y[k] = i\};$$

- the set $A_i(a)$ of cases with attribute a within values of class i is computed using:

$$\begin{aligned} A_i(a) &= \{X[k] \in X : min(C_i(a)) \leq X[k][a] \leq max(C_i(a))\} && \text{for numeric attributes,} \\ A_i(a) &= \{X[k] \in X : X[k][a] \in C_i(a)\} && \text{for nominal attributes,} \end{aligned}$$

  with the definition of $A_i(a)$ for nominal attributes adapted by us from the intuition of the definition for numeric attributes;

- the set $B_i(a)$ of cases with attribute a within values of class i but not any other class is computed using:

$$B_i(a) = A_i(a) - \cup_{i \neq j, j \in classes} A_j(a);$$

- the weight w_a for attribute a corresponds to its average "ability to discriminate", and is computing using:

$$w_a = |\cup_{i \in classes} B_i(a)|/n, \ n : len(X),$$

  and normalized as:

$$w^*_a = w_a / (\sum_{a'} w_{a'}).$$

Other approaches. We compare CoAT with multiple variants of k-NN using the same similarity measures as CoAT. The variants we consider are k-NN using a weighted voting strategy for the outcome, where each retrieved case (s_j, r_j) carries a weight in favor r_j inversely proportional with its similarity with the target situation: $w_j = 1/\sigma_{\mathcal{S}}(s_j, s_t)$. In other words, similar cases will be more important than less similar cases when deciding the outcome of a case. We perform the prediction considering the $k \in \{1, 5, 10\}$ most similar cases, as well as considering all the cases in the case base.

B.1.2 Results

In Figure B.1, we report the preliminary results for 4 of the datasets chosen for their representativeness in terms of features and amount of samples. These datasets are Iris (moderate amount of samples, few numerical features), Ionosphere (numerous samples and numerical features), Lung Cancer (few samples, numerous nominal features), and Hepatitis (moderate amount of samples, heterogeneous features).

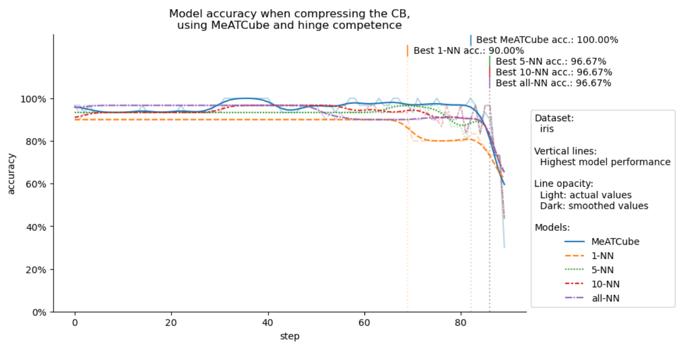
While the trajectory of the performance of CoAT is significantly more noisy than in our experiments with synthetic data, we observe that on all datasets, CoAT follows the same tendency of a curve split in three phases. It appears that only the difficulty of the task impact the shape of the performance. For instance, we observe an absence of performance increase (phase 1) and a long plateau (phase 2) on Iris, which is known to be relatively easy for CBR system to model. In comparison, Lung Cancer is relatively hard to model due to the many features and small amount of instances, and while the initial performance of CoAT is relatively low, the compression

From these preliminary experiments, it appears that k-NN does not follow the same tendency as CoAT, when performing compression with C_{hinge}. However, the evolution of the performance of k-NN is chaotic at best when considering all the datasets. Additionally, results from [Bad20] indicated a significant correlation between the performance of k-NN and the value of the Γ indicator.

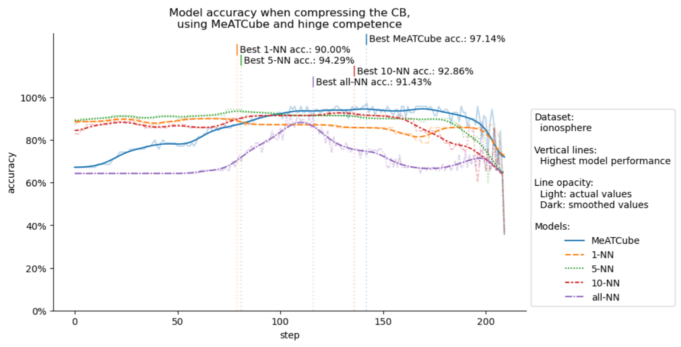
B.1.3 Limitations

In these preliminary experiments, we used only a single split for each dataset so the observed performance might not be representative, in particular for some datasets with few instances such as Lung Cancer. Using proper k-fold validation would produce more representative and less noisy results.

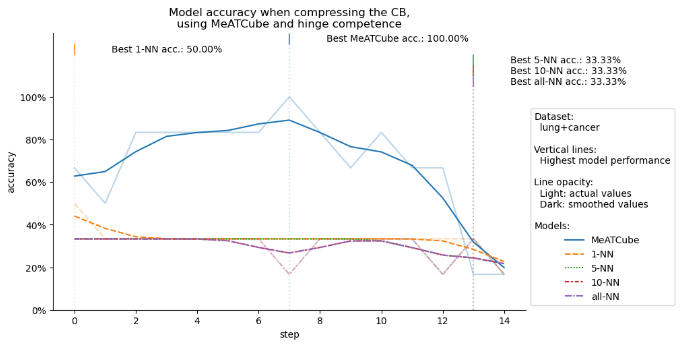
Additionally, the procedure reproduced from [KAÜ19] to determine the weights of the weighted similarity in the situation space does not appear to produce the expected performance for k-NN. This is likely a problem of either our implementation of the procedure or our choice of similarities for each features.



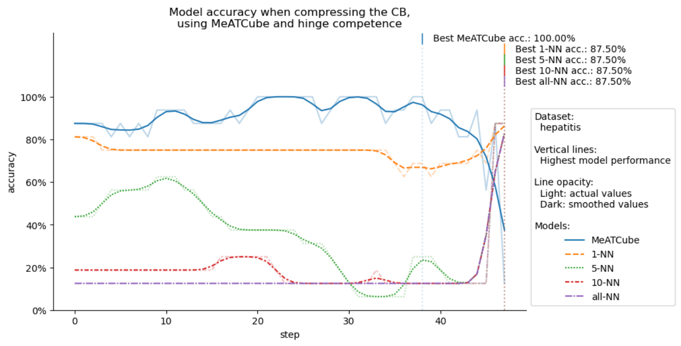
(a) Accuracy for the Iris dataset



(b) Accuracy for the Ionosphere dataset



(c) Accuracy for the Lung Cancer dataset



(d) Accuracy for the Hepatitis dataset

Figure B.1: Evolution of the accuracy during the case deletion procedure during our preliminary experiments. Smoothing has been applied to the curves to allow better reading of tendencies.

B.2 MeATCube implementation

The order of magnitude of computing a prediction with CoAT, without optimization, is of the order of magnitude of $|CB|^3|\mathcal{R}|$ and computing competence is of the order of $|CB|^4|\mathcal{T}||\mathcal{R}|$, assuming similarities have already been computed between all cases. This computational cost of the operations can be prohibitive, in particular if any of $CB, \mathcal{T}, \mathcal{R}$ is large.

To be able to run our experiments where we repeat a number of computations of case (base) competence and case influence, we implemented a reformulation of CoAT using high dimensional tensors in PyTorch. This allowed us to combine the optimization principle from [Bad+22], which already allowed for an optimization of the speed of CoAT by an order of magnitude, with the efficiency of matrix representation of Numpy and the parallelization capabilities the operations implemented in PyTorch. The resulting implementation, named MeATCube, offered a speedup of multiple orders of magnitude compared to both the naive and the optimized formulations from [Bad+22].

This appendix describes the intuition behind the implementation of MeATCube, which is available on GitHub[1]. The choice of the name MeATCube, beyond the meaning of the words we used, comes in part from MeAT, an alternative name proposed for the Γ indicator. From there, MeATCube was the solution to the analogical equation "*ball*" : "*meat ball*" :: "*cube*" : x, based on the boolean cubes used to compute Γ in MeATCube.

B.2.1 Optimization principles

Previous optimizations. Several optimizations were proposed by Badra, Lesot, et al. [Bad+22] for CoAT, with regards to the number of operations to perform to compute the contribution of a case to Γ. These optimizations bring the time complexity of CoAT from $O(|CB|^3|\mathcal{R}|)$ down to $O(|CB|^2|\mathcal{R}|)$. While we do not use the same demonstration, the results of the demonstration in [Bad+22] and the one down below are equivalent.

Rewriting Γ with regards to a given case. We can decompose $\Gamma(\sigma_\mathcal{S}, \sigma_\mathcal{R}, CB \cup \{c\})$ into all possible configurations with regards to $c \notin CB$, with $inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c_a, c_b, c_c)$ the inversion condition for cases c_a, c_b, c_c:

$$inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, (s_0, r_0), (s_i, r_i), (s_j, r_j)) = (\sigma_\mathcal{S}(s_0, s_i) \geq \sigma_\mathcal{S}(s_0, s_j)) \ \wedge \ (\sigma_\mathcal{R}(r_0, r_i) < \sigma_\mathcal{R}(r_0, r_j))$$

$$\begin{aligned}
\Gamma(\sigma_\mathcal{S}, \sigma_\mathcal{R}, CB \cup \{c\}) = &\sum_{c_a, c_b, c_c \in CB} (1 \text{ if } inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c_a, c_b, c_c) \text{ else } 0) && \text{(B.2)}\\
&+ \sum_{c_a, c_b \in CB} (1 \text{ if } inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c_a, c_b, c) \text{ else } 0) && \text{(B.3)}\\
&+ \sum_{c_a, c_c \in CB} (1 \text{ if } inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c_a, c, c_c) \text{ else } 0) && \text{(B.4)}\\
&+ \sum_{c_b, c_c \in CB} (1 \text{ if } inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c, c_b, c_c) \text{ else } 0) && \text{(B.5)}\\
&+ \sum_{c_a \in CB} (1 \text{ if } inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c_a, c, c) \text{ else } 0) && \text{(B.6)}\\
&+ \sum_{c_b \in CB} (1 \text{ if } inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c, c_b, c) \text{ else } 0) && \text{(B.7)}\\
&+ \sum_{c_c \in CB} (1 \text{ if } inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c, c, c_c) \text{ else } 0) && \text{(B.8)}\\
&+ (1 \text{ if } inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c, c, c) \text{ else } 0). && \text{(B.9)}
\end{aligned}$$

A number of these terms can be simplified (we use the logical notations $\top$ for true and $\bot$ for false):

- line (B.2) is another formulation of $\Gamma(\sigma_\mathcal{S}, \sigma_\mathcal{R}, CB)$;
- line (B.9) is always 0: for any case $c = (s, r)$, we have that $\sigma_\mathcal{R}(r, r) < \sigma_\mathcal{R}(r, r) = \bot$, therefore $inv(\sigma_\mathcal{S}, \sigma_\mathcal{R}, c, c, c) = \bot$;

[1] https://github.com/EMarquer/MeATCube

- line (B.6) is always 0: for any two cases $c = (s, r), c_a = (s_a, r_a)$, we have that $\sigma_{\mathcal{R}}(r_a, r) < \sigma_{\mathcal{R}}(r_a, r) = \bot$, therefore $inv(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c_a, c, c) = \bot$;
- line (B.8) is always 0: for any two cases $c = (s, r), c_c = (s_c, r_c)$, we have that $\sigma_{\mathcal{R}}(r, r) < \sigma_{\mathcal{R}}(r, r_c) = \bot$ as by definition $\sigma_{\mathcal{R}}(r, r) \geq \sigma_{\mathcal{R}}(r, r_c)$, therefore $inv(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c, c, c_c) = \bot$.

We can now rewrite $\Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{c\})$ using $\Gamma_c(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{c\})$, the latter being the inversions involving c, which can also be simplified:

$$\Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{c\}) = \Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB) + \Gamma_c(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{c\}) \tag{B.10}$$

$$\begin{aligned}
\Gamma_c(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{c\}) &= |\{(c_0, c_i, c_j) \in CB^3 \text{ such that } c \in \{c_0, c_i, c_j\} \wedge inv(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c_0, c_i, c_j)\}| \\
&= \sum_{c_a, c_b \in CB} (1 \text{ if } inv(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c_a, c_b, c) \text{ else } 0) \\
&+ \sum_{c_a, c_c \in CB} (1 \text{ if } inv(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c_a, c, c_c) \text{ else } 0) \\
&+ \sum_{c_b, c_c \in CB} (1 \text{ if } inv(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c, c_b, c_c) \text{ else } 0) \\
&+ \sum_{c_b \in CB} (1 \text{ if } inv(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c, c_b, c) \text{ else } 0).
\end{aligned} \tag{B.11}$$

Note that in [Bad+22, Algorithm 4], line (B.7) is also considered null, most likely due to the assumption that a given source s is associated with a single outcome r. Line (B.7) can be split in two cases:

- $\sigma_{\mathcal{S}}(s, s_b) = \sigma_{\mathcal{S}}(s, s)$, and by the above assumption, $r = r_i$ which results in $\sigma_{\mathcal{R}}(r, r_b) = \sigma_{\mathcal{R}}(r, r)$ and in turn $inv(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c, c_b, c) = \bot$;
- $\sigma_{\mathcal{S}}(s, s_b) < \sigma_{\mathcal{S}}(s, s)$ and $inv(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c, c_b, c) = \bot$.

Rewriting the optimization problem. When performing predictions with CoAT, we need to solve Equation (12.2), which can be rewritten using Equation (B.10):

$$\begin{aligned}
r_t &= \operatorname*{argmin}_{r \in \mathcal{R}} \Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, r)\}) \\
&= \operatorname*{argmin}_{r \in \mathcal{R}} \Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB) + \Gamma_{(s_t, r)}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, r)\}).
\end{aligned}$$

All triples $((s_0, r_0), (s_i, r_i), (s_j, r_j)) \in (CB \cup \{(s_t, r)\})^3$ that do not involve the new situation are common to all candidates $(s_t, r), r \in \mathcal{R}$, and their value will not impact the optimization problem. In other words, $\Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB)$ will be the same for all $r \in \mathcal{R}$ and we can rewrite the optimization problem as:

$$r_t = \operatorname*{argmin}_{r \in \mathcal{R}} \Gamma_{(s_t, r)}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, r)\}).$$

Contrary to $\Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup c)$ for which the number of computations of inv is $|CB \cup c|^3$, for $\Gamma_c(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup c)$ it is down to $3|CB|^2 + |CB|$: the non-zero lines (B.3), (B.4), and (B.5) are all sums over $|CB|^2$ terms, and (B.7) is a sum over $|CB|$ terms. As a conclusion, we indeed reduce the time complexity for the prediction with CoAT from $O(|CB|^3|\mathcal{R}|)$ to $O(|CB|^2|\mathcal{R}|)$.

Rewriting the competence. In a similar way as the optimization problem above, we can rewrite the difference in energies ℓ_{MCE} (see Equation (12.8)) as:

$$\begin{aligned}
\ell_{MCE}(\theta, c_t) &= E_\theta(s_t, r_t) - \min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} E_\theta(s_t, \overline{r}) \\
&= \Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, r_t)\}) - \min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} \Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, \overline{r})\}) \\
&= \Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB) + \Gamma_{(s_t, r_t)}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, r_t)\}) \\
&\quad - \min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} (\Gamma(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB) + \Gamma_{(s_t, \overline{r})}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, \overline{r})\})) \\
&= \Gamma_{(s_t, r_t)}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, r_t)\}) - \min_{\overline{r} \in \mathcal{R},\, \overline{r} \neq r_t} \Gamma_{(s_t, \overline{r})}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{(s_t, \overline{r})\}),
\end{aligned}$$

which depends only on Γ_c and benefits from a time complexity in $O(|CB|^2|\mathcal{R}|)$ instead of $O(|CB|^3|\mathcal{R}|)$. This new definition impacts ℓ_{hinge}, but also every case (base) competence and case influence notion we defined before: $O(\ell_{hinge}) = O(|CB|^2|\mathcal{R}|)$, and $O(C_{MCE}) = O(C_{hinge}) = O(|CB|^2|\mathcal{R}|\ |\mathcal{T}|)$.

B.2.2 Boolean cube reformulation of CoAT: MeATCube

The representation of matrices in PyTorch offers many optimizations in terms of space and time complexity. The MeATCube implementation we design is entirely written in PyTorch, so it can be used to perform most of the computations on GPU. As such, it is possible to benefit from the parallelization capabilities the operations implemented in PyTorch, and achieve significant speedups.

Basic operations and broadcast mechanism. In all the definition below we use the usual indexing in PyTorch that start at 0.

We consider a similarity matrix such that coordinates match that way: $\sigma[i,j] = \sigma(i,j)$. Following PyTorch notation, the slice $\sigma[i,:]$ of σ is the vector where coordinate j' corresponds $\sigma(i,j')$. Similarly, $\sigma[:,j]$ is the vector containing $\sigma(i',j)\forall i'$.

We use the component-wise operators: $\&, \geq, <$ for the component-wise $\wedge, \geq, <$. We also use $\sum(A)$ as the sum of all the values in A, and $\sum_{i,\dots}(A)$ as the sum of all the values in A along coordinates $i,\dots$. For instance, for matrix A in Example 2.1, $\sum(A) = \sum_{0,1}(A) = 21$, $\sum_0(A) = [1+3+5, 2+4+6] = [9,12]$, and $\sum_1(A) = [3,7,11]$. We define $\max_{i,\dots}(A), \min_{i,\dots}(A), \prod_{i,\dots}(A)$ in the same manner. We also define $\text{argmax}_i(A)$ as the index that maximizes A along the coordinate i, all other dimensions preserved, for instance $\text{argmax}_0(A) = [2,2]$ and $\text{argmax}_1(A) = [1,1,1]$.

We use the automatic *broadcasting* of PyTorch in our writing. It works as follows: for matrices A, B, C of size $M \times N$, $1 \times N$, and $M \times 1$ respectively, if we apply a component-wise operator between any two of them, broadcasting will make them have the same size $M \times N$ by copying the values along the coordinate of size 1. For instance, for $A \wedge B$, B will be replaced by B' of size $M \times \mathbf{N}$ obtained by copying B along the second coordinate N times. For $B < C$, both B and C are broadcast.

To enable broadcasting, we use the unsqueeze operator $\mathit{unsqueze}(A,k)$ to add a coordinate of size 1 to a matrix or vector A as the k-th coordinate. For instance, if A is a vector of size $M \times N$, $\mathit{unsqueze}(A,0)$ will be of size $1 \times M \times N$, $\mathit{unsqueze}(A,1)$ will be of size $M \times 1 \times N$, and $\mathit{unsqueze}(A,2)$ will be of size $M \times N \times 1$.

The important advantage of broadcasting over an explicit copy of the values is that the values are not actually copied when broadcasting, which saves significant computation time and memory: we do not need to store the copied values in memory, which also saves us the time required to reserve memory space and affect the values. Using slices in PyTorch follows similar principles, with the original data not copied until explicitly required by the user. As an example, defining a slice $A' = A[:,1]$ of a matrix A is not more than defining a set of coordinates to mask A when reading A'. In this example, we take all the values along the first coordinate, and we take the second value along the second coordinate ($i = 1$, starting from $i = 0$).

The broadcasting and unsqueezing mechanisms can be combined to perform the equivalent of parallel computations.

Example 2.1: Broadcasting and unsqueezing

Let us take the following A, B:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 3.5 & 4 \end{bmatrix}.$$

Let us assume we want to check that every component in the first row of A is strictly inferior to the first component of B, every component in the second row of A to the second component of B, and so on. To do so, we can not compute the component-wise $A < B$, as A, B do not have the same size (respectively 3×2 and 3). Using broadcasting and unsqueezing, we can nevertheless compute $A < \mathit{unsqueze}(B,1)$, by first adding a phantom coordinate to B, with $\mathit{unsqueze}(B,1)$ of size 3×1, then broadcasting the result to the size of A ($\top$: true, $\perp$: false):

$$A < \mathit{unsqueze}(B,1) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} < \begin{bmatrix} 3 & 3 \\ 3.5 & 3.5 \\ 4 & 4 \end{bmatrix} = \begin{bmatrix} \top & \top \\ \top & \perp \\ \perp & \perp \end{bmatrix}.$$

Cube of inversions and indicator Γ. The inversions required for Γ correspond to all inversion considering triplets of elements in CB^3, that we can represent by a cube containing boolean values. We will explain step by step this process.

Let us consider the similarity matrices $\sigma_{\mathcal{S}}$ and $\sigma_{\mathcal{R}}$ such that $\sigma_{\mathcal{S}}[a,b]$ is the similarity $\sigma_{\mathcal{S}}(s_a, s_b)$ and $\sigma_{\mathcal{R}}[a,b]$ is the similarity $\sigma_{\mathcal{R}}(r_a, r_b)$ for two cases c_a, c_b indexed by $a, b \in [0, |CB|-1]$. We can represent every comparisons $\sigma_{\mathcal{S}}(s_a, s_b) \geq \sigma_{\mathcal{S}}(s_a, s_c)$ using the cube $\sigma_{\mathcal{S}\geq} = \mathit{unsqueeze}(\sigma_{\mathcal{S}}, 2) \geq \mathit{unsqueeze}(\sigma_{\mathcal{S}}, 1)$ which is such that $\sigma_{\mathcal{S}\geq}[a,b,c] = \sigma_{\mathcal{S}}(s_a, s_b) \geq \sigma_{\mathcal{S}}(s_a, s_c)$. We explain in more detail, for $a, b, c \in [0, |CB|-1]$:

- $\mathit{unsqueeze}(\sigma_{\mathcal{S}}, 2)$ is of size $|CB| \times |CB| \times 1$, and will be copied along the new third coordinate; as such, $\mathit{unsqueeze}(\sigma_{\mathcal{S}}, 2)[a,b,c] = \sigma_{\mathcal{S}}(s_a, s_b)$;
- $\mathit{unsqueeze}(\sigma_{\mathcal{S}}, 1)$ is of size $|CB| \times 1 \times |CB|$, and will be copied along the new second coordinate; as such, $\mathit{unsqueeze}(\sigma_{\mathcal{S}}, 1)[a,b,c] = \sigma_{\mathcal{S}}(s_a, s_c)$;
- from this, $\mathit{unsqueeze}(\sigma_{\mathcal{S}}, 2) \geq \mathit{unsqueeze}(\sigma_{\mathcal{S}}, 1)$ broadcasts to the same size $|CB| \times |CB| \times |CB|$ and we have:

$$\begin{aligned}\sigma_{\mathcal{S}\geq}[a,b,c] &= (\mathit{unsqueeze}(\sigma_{\mathcal{S}}, 2) \geq \mathit{unsqueeze}(\sigma_{\mathcal{S}}, 1))[a,b,c] \\ &= \mathit{unsqueeze}(\sigma_{\mathcal{S}}, 2)[a,b,c] \geq \mathit{unsqueeze}(\sigma_{\mathcal{S}}, 1)[a,b,c] \\ &= \sigma_{\mathcal{S}}(s_a, s_b) \geq \sigma_{\mathcal{S}}(s_a, s_c).\end{aligned}$$

In the same way, we can define $\sigma_{\mathcal{R}<} = \mathit{unsqueeze}(\sigma_{\mathcal{R}}, 2) < \mathit{unsqueeze}(\sigma_{\mathcal{R}}, 1)$ which is such that $\sigma_{\mathcal{R}<}[a,b,c] = \sigma_{\mathcal{R}}(s_a, s_b) < \sigma_{\mathcal{R}}(s_a, s_c)$.

From these two boolean cubes of size $|CB|^3$, we can compute a final boolean cube $\mathit{inv} = \sigma_{\mathcal{S}\geq} \,\&\, \sigma_{\mathcal{R}<}$, which is such that $\mathit{inv}[a,b,c] = \mathit{inv}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, c_a, c_b, c_c)$. As PyTorch uses the standard encoding of boolean as bits, we have $\top = 1, \bot = 0$, and the value of Γ is the sum of all values in the inversion cube inv, *i.e.*, $\Gamma = \sum \mathit{inv}$.

Contribution of a case to the inversion cube. Using the principle in Appendix B.2.1, we do not need to compute the whole cube in most cases. To compute the contribution of a new case c_i to the value of CoAT, we need to compute only a subset of values according to Equation (B.11):

$$\begin{aligned}\Gamma_{c_i}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{c_i\}) = &\sum_{a,b \in [0,|CB|-1]} \mathit{inv}[a,b,i] \\ &+ \sum_{a,c \in [0,|CB|-1]} \mathit{inv}[a,i,c] \\ &+ \sum_{b,c \in [0,|CB|-1]} \mathit{inv}[i,b,c] \\ &+ \sum_{b \in [0,|CB|-1]} \mathit{inv}[i,b,i]\end{aligned}$$

These values correspond to slices of the cube, where the value of one (for $\mathit{inv}[a,b,i], \mathit{inv}[a,i,c], \mathit{inv}[i,b,c]$) or two coordinates (for $\mathit{inv}[i,b,i]$) are fixed. The values in those slices can be obtained without computing the full inversion cube if we have $\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}$ the pairwise similarity matrices within CB (of size $|CB|^2$), as well as $\sigma_{\mathcal{S}i}, \sigma_{\mathcal{R}i}$ (of size $|CB|$) the similarity vectors between c_i and every case in CB. The formulas are as follow, using constant values $\sigma_{\mathcal{S}ii}, \sigma_{\mathcal{R}ii}$ for the similarity between a case and itself:

- to obtain the values corresponding to $\mathit{inv}[a,b,i]$ for $a, b \in [0, |CB|-1]$: $\mathit{inv}_{abi} = \sum_{0,1}((\sigma_{\mathcal{S}} \geq \mathit{unsqueeze}(\sigma_{\mathcal{S}i}, 1)) \& (\sigma_{\mathcal{R}} < \mathit{unsqueeze}(\sigma_{\mathcal{R}i}, 1)))$, which results in a $|CB| \times |CB|$ matrix for which we take the sum;
- to obtain the values corresponding to $\mathit{inv}[a,i,c]$ for $a, c \in [0, |CB|-1]$: $\mathit{inv}_{aic} = \sum_{0,1}((\mathit{unsqueeze}(\sigma_{\mathcal{S}i}, 1) \geq \sigma_{\mathcal{S}}) \& (\mathit{unsqueeze}(\sigma_{\mathcal{R}i}, 1) < \sigma_{\mathcal{R}}))$, which results in a $|CB| \times |CB|$ matrix for which we take the sum;
- to obtain the values corresponding to $\mathit{inv}[i,b,c]$ for $b, c \in [0, |CB|-1]$: $\mathit{inv}_{ibc} = \sum_{0,1}((\mathit{unsqueeze}(\sigma_{\mathcal{S}i}, 2) \geq \mathit{unsqueeze}(\sigma_{\mathcal{S}i}, 1)) \& (\mathit{unsqueeze}(\sigma_{\mathcal{R}i}, 2) < \mathit{unsqueeze}(\sigma_{\mathcal{R}i}, 1)))$, which results in a $|CB| \times |CB|$ matrix for which we take the sum;

- to obtain the values corresponding to $inv[i, b, i]$ for $b \in [0, |CB| - 1]$:
  $inv_{ibi} = \sum_0((\sigma_{\mathcal{S}i} \geq unsqueeze(\sigma_{\mathcal{S}ii}, 0))\&(\sigma_{\mathcal{R}i} < unsqueeze(\sigma_{\mathcal{R}ii}, 0)))$, which results in a vector of size $|CB|$ for which we take the sum, with $unsqueeze(x, 0)$ the transformation of the value x into the vector $[x]$.

We can then sum everything to obtain the formulation of MeATCube:

$$\begin{aligned} \text{MeATCube}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, \sigma_{\mathcal{S}i}, \sigma_{\mathcal{R}i}, \sigma_{\mathcal{S}ii}, \sigma_{\mathcal{R}ii}) &= inv_{abi} + inv_{aic} + inv_{ibc} + inv_{ibi} \qquad \text{(B.12)} \\ &= \Gamma_{c_i}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{c_i\}). \qquad \text{(B.13)} \end{aligned}$$

Note that in the actual implementation of MeATCube, the coordinates for the unsqueeze operations are automatically adapted to fit any number of extra dimensions. The matrices are defined as:

- $\sigma_{\mathcal{S}}[\ldots, a, b] = \sigma_{\mathcal{S}}(a, b) \forall a, b \in CB$, of size $|CB| \times |CB|$ for the last two dimensions;
- $\sigma_{\mathcal{R}}[\ldots, a, b] = \sigma_{\mathcal{R}}(a, b) \forall a, b \in CB$, of size $|CB| \times |CB|$ for the last two dimensions;
- $\sigma_{\mathcal{S}i}[\ldots, a] = \sigma_{\mathcal{S}}(a, i) \forall a \in CB$, of size $|CB|$ for the last dimension;
- $\sigma_{\mathcal{R}i}[\ldots, a] = \sigma_{\mathcal{R}}(a, i) \forall a \in CB$, of size $|CB|$ for the last dimension;
- $\sigma_{\mathcal{S}ii}[\ldots] = \sigma_{\mathcal{S}}(i, i)$;
- $\sigma_{\mathcal{R}ii}[\ldots] = \sigma_{\mathcal{R}}(i, i)$.

Advanced operations The contribution of a new case c to the value of Γ if it was added to the case base can be generalized. The principle of this generalization is based on the broadcast mechanism.

For instance, suppose we want to perform a prediction. We need to find the outcome r_t that minimizes the value of $\Gamma_{c_i}(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, CB \cup \{c_i\})$. With MeATCube, this can be done naively by repeating MeATCube$(\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, \sigma_{\mathcal{S}i}, \sigma_{\mathcal{R}i}, \sigma_{\mathcal{S}ii}, \sigma_{\mathcal{R}ii})$ for all possible $r_t \in \mathcal{R}$. However, $\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, \sigma_{\mathcal{S}i}, \sigma_{\mathcal{S}ii}$, and most likely $\sigma_{\mathcal{R}ii}$ will not change for different values of $r_j \in \mathcal{R}$, so it would be a waste to recompute the terms that do not depend on $\sigma_{\mathcal{R}i}$ each time. This can be done with $\sigma_{\mathcal{R}\mathcal{R}i}$, a new matrix for which $\sigma_{\mathcal{R}\mathcal{R}i}[j] = \sigma_{\mathcal{R}i}$ for $r_j \in \mathcal{R}$. We can then compute the index t of the best outcome in $r_t \in \mathcal{R}$:

$$\begin{aligned} t = \operatorname{argmin} \text{MeATCube}(&unsqueeze(\sigma_{\mathcal{S}}, 0), unsqueeze(\sigma_{\mathcal{R}}, 0), unsqueeze(\sigma_{\mathcal{S}i}, 0), \sigma_{\mathcal{R}\mathcal{R}i}, \\ &unsqueeze(\sigma_{\mathcal{S}ii}, 0), unsqueeze(\sigma_{\mathcal{R}ii}, 0)). \end{aligned}$$

In general, each time we want to repeat a computation of MeATCube for different variants of a value $\alpha \in \mathbf{A}$, we add a new coordinate, which will be a phantom coordinate of size 1 for matrices that do not change for different α. For matrices that depend on α, the coordinate will have a size of $|\mathbf{A}|$ and contain the different variants of the matrix, as we did for $\sigma_{\mathcal{R}i}$ by producing $\sigma_{\mathcal{R}i\mathcal{R}}$.

B.2.3 Observed speed on real-world datasets.

In Figure B.2, we report the time required to perform a single deletion step in our experiments from Appendix B.1. As a reminder, the time complexity of MeATCube for competence is $O(|CB|^2|\mathcal{R}|\ |\mathcal{T}|)$. In our experiments the reference set is proportional to the initial size of the case base, and the set of possible outcomes does not change through executions. As such, $|\mathcal{R}|$ and $|\mathcal{T}|$ can be seen as constants depending on the dataset during the experiment. We observe that the time complexity is proportional to $|CB|^2$, which is consistent with the above, while differences in slopes between the datasets can be attributed to $|\mathcal{R}|$ and $|\mathcal{T}|$.

We did not have an implementation of the naive implementation nor the optimization from Badra, Lesot, et al. [Bad+22] available, however we can compare the orders of magnitudes with estimates. In [Bad+22], a formula of the run time is available for the computation of CoAT, which is equivalent to the computation of ℓ_{MCE} in our implementation, as the energy (*i.e.*, Γ) is computed once per possible outcome. As can be read in Equations (12.12) and (12.13), in each compression step we compute ℓ_{MCE} twice to obtain the influence, that we then average over $\mathcal{T}$. In our experiments we used a reference of a size $|\mathcal{T}| = |CB|/3$, so we need to multiply the value of the

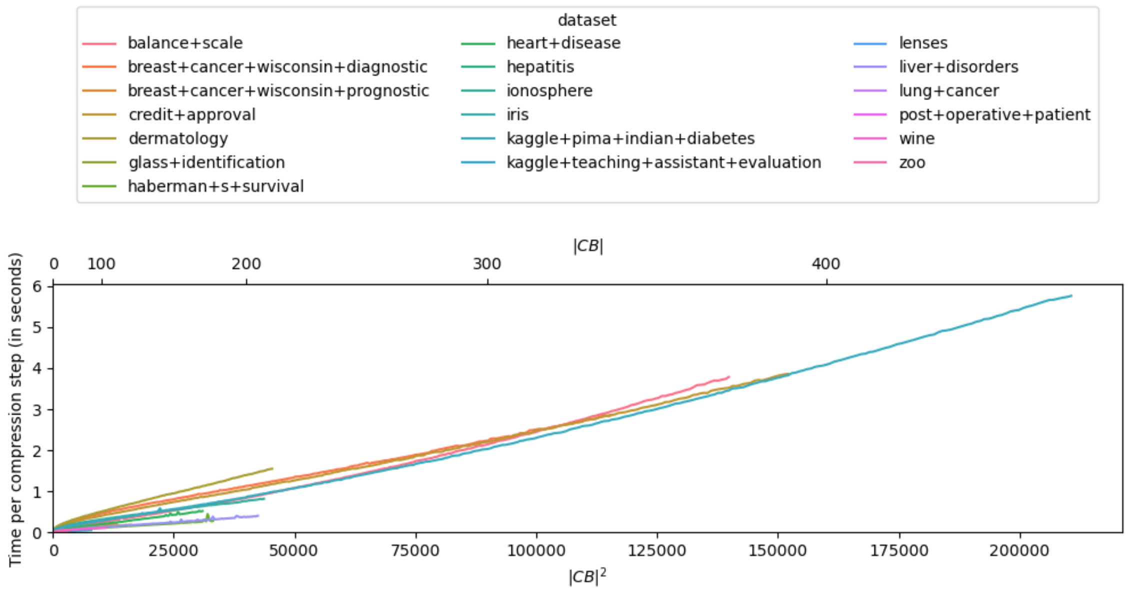


Figure B.2: Runtime observe (in seconds) for each step of the compression process, on an Intel Xeon W-11955M CPU@2.60GHz. Each compression step corresponds to computing the competence $C_{hinge}(c, \theta, \mathcal{T})$ of all cases c for all the references in $\mathcal{T}$, averaging over the references, finding the case with the least case competence nad removing it.

estimate by $2|CB|/3$. For $|CB| = 300$ we estimate a time of 138 seconds for the optimized version, and 1002 seconds for the non-optimized version. According to Figure B.2, we need around 2 seconds with MeATCube on the Balance Scale dataset used in [Bad+22]. The values for the naive and optimized CoAT are estimates based on results obtained with unknown hardware, so they should not be taken at face value. However, they allow us to confirm that we are two orders of magnitude faster than the optimized implementation from [Bad+22], and three orders of magnitude faster than the naive implementation, with in addition a cubic growth for the naive implementation and a quadratic one for the optimized CoAT and MeATCube.

Appendix C

Axiomatic analogy beyond APs: towards a systematic data augmentation procedure

Chapter contents

C.1 Introduction

APs are a specific form of analogy that follows formal rules. One way to define APs is, as was done in [CL24; Lep01; LA96; PR18, for instance], by using postulates to form an axiomatic system. From the axiomatic system of APs, it is possible to perform data augmentation to train machine learning models on analogical data (see Section 6.3). However, some approaches, for instance [Als+22b;

Ant22; CL24, see also Subsection 2.3.2], discuss the relevance of specific postulates and consider models that do not fit all the postulates of APs.

This document proposes to extend and generalize the data augmentation procedure used in Section 6.3 and Chapters 10 and 11 on APs, motivated by the success of [MMC22, reported in Section 7.6]. The generalized procedure is theoretically grounded in the postulates of APs, is modular and actionable. Indeed, the building blocks are straightforward generalizations of the postulates of APs, that can be combined to fit various models related to analogy. The impact of the postulates, of the assumptions on the data, and of the heuristics used is made clear in Appendix C.3. In this mindset, the current proposal describes links with the properties of the relations underlying the analogy in Appendix C.4, as a first guideline for implementing the proposed analogical data augmentation framework to real use cases.

Before describing the generalized data augmentation process, we discuss the links between different subsets of postulates and with existing (Boolean) models of analogies between quadruples in Appendix C.2.

In order to avoid confusions as much as possible, in this section we use different terms:

- we call APs the quadruples following the 8 forms of Lepage;
- we call quadruples or permutations the ones that do not necessarily following the 8 forms;
- in general, an analogical or non-analogical quadruple from a dataset is called an exemplar.

C.2 Hierarchy of models of analogy underlying subsets of postulates

C.2.1 Reminder on the categories of postulates

As mentioned in Appendix C.1, APs can be defined using postulates in an axiomatic setting, that we separate into 3 categories described in Subsection 2.3.3 and recalled bellow:

- **permutation postulates:** postulates that state that if an AP $A : B :: C : D$ exists, then another AP exists, obtainable by permuting A, B, C, D;
- **existence postulates:** postulates that state that a particular analogical form exists;
- **constraint postulates:** postulates that constrain the possible values of an analogical form.

The advantages of this distinction come to light when we consider data augmentation, in which permutation postulates play a central role. The permutation postulates and their associated category are detailed in Table 2.1, that we recall in Table C.1. Some of the permutation postulates can be obtained by combining other permutation postulates and are usually not considered on their own. To the best of our knowledge, these permutation postulates do not have frequently used names, so in Table C.1 we propose the placeholder names of extra postulate 1 and extra postulate 2. As mentioned in Subsection 2.3.2, in certain setting it makes sense to consider a Symmetry of Ratio postulate.

Additional permutation postulates. While we limit ourselves to a subset of permutation postulates that, in our opinion, can be explained intuitively, our framework can in theory accommodate any of the 24[1] permutations of A, B, C, D as permutation postulate.

Additional existence postulates and constraint postulates. Recently, a discussion on the Boolean models of AP was proposed by Leemhuis and Özçep [LÖ23]. In their work, they proposed several existence postulates, defined for all A, B, C, D and some fixed k[2]:

$$A : B :: B : A \qquad \text{(ratio symmetry)}$$
$$A : A :: B : C \qquad \text{(universal neutrality)}$$
$$k : k :: A : B \qquad \text{(k-neutrality)}$$
$$A : B :: C : D, \qquad \text{(universality)}$$

[1]Excluding $A : B :: C : D \implies A : B :: C : D$ that is always true and might cause infinite loops depending on the implementation of our approach.

[2]We propose to generalize the definitions from [LÖ23] by using k instead of specific Boolean values 0 and 1.

Postulate	Symbol	Description
		Permutation postulates group
Symmetry of Conformity	*SymC*	$A : B :: C : D \implies C : D :: A : B$
Exchange of the Means	*EM*	$A : B :: C : D \implies A : C :: B : D$
Inversion of Ratio	*IR*	$A : B :: C : D \implies B : A :: D : C$
Exchange of the Extremes	*EE*	$A : B :: C : D \implies D : B :: C : A$
Symmetry of Reading	*Rev*	$A : B :: C : D \implies D : C :: B : A$
Extra postulate 1	*Ex*1	$A : B :: C : D \implies C : A :: D : B$
Extra postulate 2	*Ex*2	$A : B :: C : D \implies B : D :: A : C$
*Symmetry of Ratio	*Sym* :: *Sym*	
**Symmetry of Left Ratio*	*Sym* ::	$A : B :: C : D \implies B : A :: C : D$
**Symmetry of Right Ratio*	:: *Sym*	$A : B :: C : D \implies A : B :: D : C$
		Existence postulates group
Reflexivity of Conformity	*Ref*	$A : B :: A : B$
Identity	*Id*	$A : A :: B : B$
Solvability	*Solv*	$\forall A, B, C\ \exists D$ such that $A : B :: C : D$ holds
		Constraint postulates group
Uniqueness	*Uniq*	$A : B :: C : D \land A : B :: C : D' \implies D = D'$
Strong Reflexivity of Conformity	*UniqRef*	$A : B :: A : D \implies B = D$, can be seen as Uniqueness restricted to Reflexivity of Conformity
Strong Identity	*UniqId*	$A : A :: C : D \implies C = D$, can be seen as Uniqueness restricted to Identity
Distribution	*Dist*	$A : B :: C : D \implies \mathcal{X}(A) \subseteq \mathcal{X}(B) \cup \mathcal{X}(C)$, with $\mathcal{X}(A), \mathcal{X}(B), \mathcal{X}(C)$ the features of A, B, C

Table C.1: Known postulates used in axiomatic analogies, excluding transitivity. Asterisks indicate permutation postulates that are not accepted for APs.

as well as corresponding constraint postulates:

$$\begin{aligned} A : B :: B : A &\implies A = B && \text{(ratio anti-symmetry)} \\ A : A :: B : C &\implies B = C && \text{(universal anti-neutrality)} \\ k : k :: A : B &\implies A = B. && (k\text{-anti-neutrality}) \end{aligned}$$

Note that universal anti-neutrality is an other name for Strong Identity. We find these postulates particularly interesting as they complement the Symmetry of Ratio postulate we propose from a different axis. For instance, the ratio symmetry proposed by Leemhuis and Özçep can be seen as Symmetry of Ratio applied to the Reflexivity of Conformity. What is interesting is that ratio symmetry of Leemhuis and Özçep generates less new quadruples than our Symmetry of Ratio, which could be useful in some use cases.

Additional categories. Other categories could be defined, for instance containing the Transitivity permutation postulate and its variants. Below we list the forms that are mentioned. In parenthesis, we provide a compressed form with a slight abuse of notation, as we chain the ratio (:) and the conformity of ratios (::):

$$A : B :: C : D \quad\land\quad C : D :: E : F \implies A : B :: E : F \qquad (A : B :: C : D :: E : F), \tag{C.1}$$
$$A : B :: C : D \quad\land\quad B : E :: D : F \implies A : E :: C : F \qquad (A : B : E :: C : D : F), \tag{C.2}$$
$$A : B :: B : C \quad\land\quad B : C :: C : D \implies A : B :: C : D \qquad (A : B :: B : C :: C : D). \tag{C.3}$$

Antić [Ant22] labels (C.1) as transitivity and (C.2) as inner transitivity. To stay consistent with the notation from Lepage, we label (C.1) as Transitivity of Conformity and (C.2) as Transitivity of Ratio. Note that the basic form of Transitivity is the one of Transitivity of Conformity, while

Transitivity of Ratio is an application of Transitivity between applications of Exchange of the Means. Antić also consider (C.3) that he calls central transitivity. In our reading, the latter is an application of the Transitivity of Conformity to $A : B :: B : C$ and $B : C :: C : D$.

Notice that the Transitivity postulate and its variants behave like existence postulates, as they state that a given quadruple is analogical. The main difference between the current existence postulates and the Transitivity postulates is that the latter is conditioned by the presence of two other quadruples as analogical exemplars. For instance, the first form of Transitivity states that $A : B :: E : F$ holds if $A : B :: C : D$ and $C : D :: E : F$ holds. However, $A : B :: E : F$ can hold without $A : B :: C : D$ and $C : D :: E : F$ holding for some C, D, therefore Transitivity is not a constraint postulate. Therefore, if we were to integrate the Transitivity postulate and its variants in our data augmentation process, they would be used in the same way as the existence postulates.

C.2.2 Equivalence classes of permutations induced by postulates.

When using permutation postulates, given an analogy $A : B :: C : D$ it is possible to generate a set of analogies obtained by permuting the elements of $A : B :: C : D$. We call that the permutations induced by the postulates. Using different sets of permutation postulates may result in different induced permutations. However, certain postulates can be deducted from other postulates, and adding them will not change the induced permutations.

Equivalence classes and cover. By repeatedly applying the postulates $p \in P$ of a set P of permutation postulates on the base form $A : B :: C : D$, we obtain a set of permutations that is stable by application of the permutation postulates. In other words, once we obtain the stable set, applying the permutation postulates $p \in P$ on any of the permutations in the set will result in a permutation already in the stable set. We call this set the equivalence class induced by the permutation postulates, written $\mathcal{C}_P$. This set is minimal: as we start from only one form, all the other forms must be reachable by a series of application of the permutation postulates P. Removing one form from the equivalence class means that at least one of the permutation postulates does not hold, and the resulting set is not stable anymore by application of all the selected permutation postulates.

It can be shown that the equivalence class $\mathcal{C}_P$ produces a cover, as was proven by Lepage [Lep03] for $\mathcal{C}_{\{SymC,EM\}}$, the set of 8 permutations we use in Part II and Chapter 2. The proof from Lepage can be generalized to any subset of permutation postulates, and the number of instances of the equivalence class that are necessary to produce the cover is $24/|\mathcal{C}_P|$.

Proof The proof can be obtained by reasoning by the absurd. Consider two distinct but overlapping instances of a equivalence class $\mathcal{C}_P$. As they overlap, they share an element, and all elements in the set are reachable by application of the permutation postulates P on any of the other elements. Therefore, all the elements of the first instance are reachable from the second instance, and conversely. In other words, the two instances must be equal, and we reach a contradiction.

Lattice of the equivalence classes. From all the permutation postulate in Table C.1, we generate all possible combinations, from $\emptyset$ to $\{SymC, EM, IR, EE, Rev, Ex1, Ex2, Sym ::, :: Sym\}$. When $Sym ::$ and $:: Sym$ appear simultaneously in the labelling, we write $Sym :: Sym$ instead. Groups of postulates with the same induced permutations form an equivalence class.

The resulting equivalence classes are displayed in Figure C.1. In the diagram, each node corresponds to an equivalence class, which may correspond to several sets of postulates. The minimal sets of postulates to obtain a given class are called its minimal generators. In the diagram, we indicate in orange the number of classes needed to cover the set of all possible permutations of A, B, C, D. This number can be derived from the size of the equivalence class and the formula $24/|\mathcal{C}_P|$ introduced in a previous paragraph. Every green colored postulate indicates the smallest equivalence class in which the postulate first appears.

Details on the equivalence classes. With $\mathcal{P}$ the set of all the permutation postulates mentioned in Table C.1, we call $\mathcal{C}_P$ equivalence class corresponding to a set of postulates $P \in \mathcal{P}$. We obtain the following equivalence classes, for which we list the minimal generators:

- the class with no postulate: $\mathcal{C}_\emptyset$

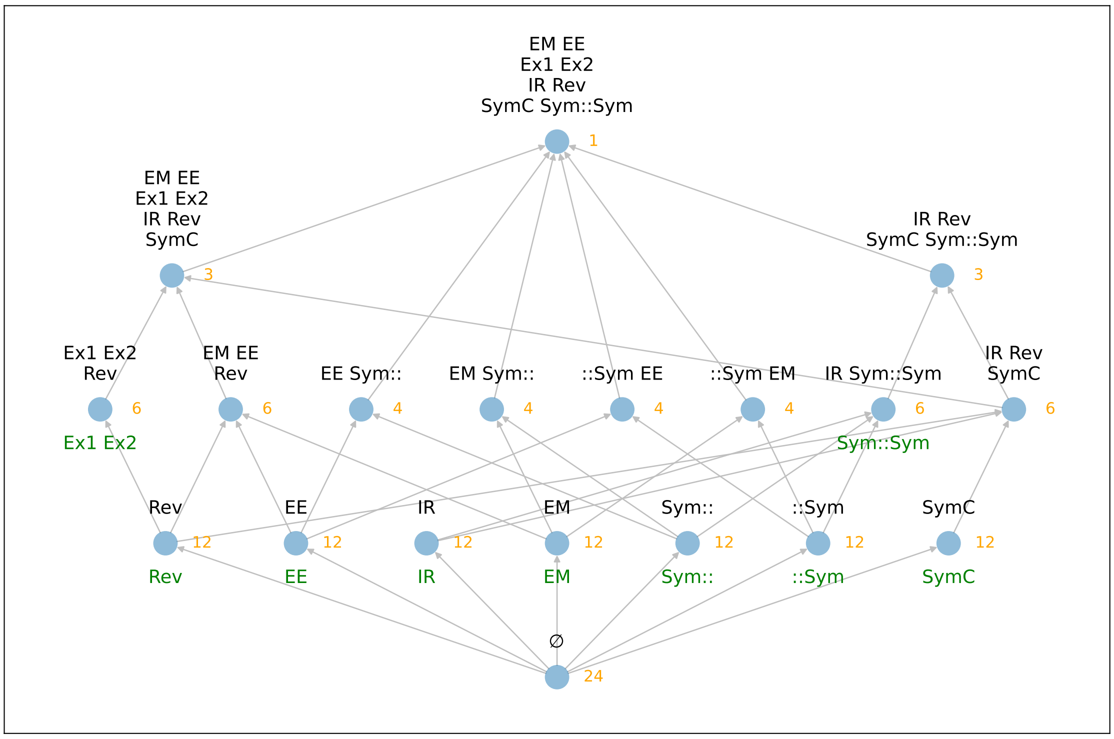


Figure C.1: Lattice containing equivalence classes induced by each group of postulates. Black is maximal set of postulates. Green is reduced class labels, that is, the equivalence class in which each postulate appears first, starting from $\perp$. Orange is number of equivalence classes to cover all permutations, that is $24/|\mathcal{C}_P|$.

- the classes with a single postulate: $\mathcal{C}_{\{SymC\}}$, $\mathcal{C}_{\{::Sym\}}$, $\mathcal{C}_{\{Sym::\}}$, $\mathcal{C}_{\{EM\}}$, $\mathcal{C}_{\{IR\}}$, $\mathcal{C}_{\{EE\}}$, and $\mathcal{C}_{\{Rev\}}$
- some classes with two postulates (excluding $\mathcal{C}_{\{Rev,Sym::Sym\}}$ as $Sym :: Sym$ is counted as two postulates): $\mathcal{C}_{\{::Sym,EM\}}$, $\mathcal{C}_{\{::Sym,EE\}}$, $\mathcal{C}_{\{EM,Sym::\}}$, $\mathcal{C}_{\{EE,Sym::\}}$; it is interesting that we can obtain equivalence classes from these postulates, even if the corresponding permutations go against most intuitions about analogy;
- $\mathcal{C}_{\{Rev,Ex1,Ex2\}}$, that can be fully generated by $Ex1$ and by $Ex2$;
- some classes with three postulates, that can be generated by any combination of 2 of their postulates: $\mathcal{C}_{\{Rev,IR,SymC\}}$, $\mathcal{C}_{\{IR,Sym::Sym\}}$, and $\mathcal{C}_{\{EM,EE,Rev\}}$;
- $\mathcal{C}_{\{IR,Rev,SymC,Sym::Sym\}}$, that can be generated by any two APs p_1, p_2:

$$(p_1, p_2) \in \{Sym ::, :: Sym\} \times \{SymC, Rev\}$$

- $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$, that is to say the 8 permutations usually accepted for APs, and can be generated by any of the subsets (p_1, p_2) in

$$(p_1, p_2) \in (G_{Ex} \times G_{SymC}) \cup (G_{SymC} \times G_{EM}) \cup (G_{Ex} \times G_{EM})$$

  with $G_{Ex} = \{Ex1, Ex2\}$, $G_{SymC} = \{IR, SymC\}$, $G_{EM} = \{EM, EE\}$;
- finally, $\mathcal{C}_{\mathcal{P}}$ is maximally equivalent to the set of all postulates, and can be generated using either:
  - 2 postulates $(p_1, p_2) \in \{Sym ::, :: Sym\} \times \{Ex1, Ex2\}$;
  - 3 distinct postulates $p_1 \neq p_2 \neq p_3 \neq p_1$ with $p_1, p_2, p_3 \in \{EM, EE, Sym ::, :: Sym\}$;

– any 3 postulates p_1, p_2, p_3:

$$(p_1, p_2, p_3) \in G_{Sym::Sym} \times G_{EM} \times G_{Rev}$$

with $G_{Sym::Sym} = \{Sym ::, :: Sym\}$, $G_{EM} = \{EM, EE\}$, $G_{Rev} = \{Rev, IR, SymC\} = \{Rev\} \cup G_{SymC}$.

Extra postulate 1 and extra postulate 2. Interestingly, removing $Ex1, Ex2$ from the considered postulates only removes equivalence classes 8. Excluding $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ and $\mathcal{C}_{\mathcal{P}\}}$, that loose some generators, no other equivalence class is significantly impacted. We find interesting to be able to define the 8 equivalent permutations of APs ($\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$) using one of $Ex1, Ex2$ instead of any of the other two postulates usually used to generate $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$.

C.2.3 Usual boolean models of analogy and the corresponding equivalence classes

In the work of Couceiro and Lehtonen [CL24], Prade and Richard [PR18], and Leemhuis and Özçep [LÖ23], among other, Boolean models of analogy are studied as a simplification of analogies on feature spaces. The idea is to focus on a particular feature, and represent the absence (0) or presence (1) of the feature in each of A, B, C, D. For instance, the R1 model (called Ω_0 in the work of Prade and Richard) is the smallest model that implements the usual postulates of APs. It is defined as:

$$\begin{matrix} A \\ B \\ C \\ D \end{matrix} \in \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \tag{C.4}$$

with each column representing a possible combination of values for $A : B :: C : D$ to hold. The model can then be generalized for any number of features, by checking if one of the columns (not necessarily the same) holds for each feature.

We consider models R1 to R8 mentioned by Couceiro and Lehtonen [CL24]. We also consider models Ω_0, Ω, and M3 to M7 mentioned in by Prade and Richard [PR18]. Two models appear in the work of Prade and Richard and of Couceiro and Lehtonen under different names: R1 and R2 are called Ω_0 and Kl by Prade and Richard.

We find the maximal equivalence class under which each model is closed. In other words, we find the largest set of postulates that are satisfied by the model. The results are as follows:

- in $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ we find R1/Ω_0 (originally defined by Miclet and Prade [MP09b]), M3, M4, and M7, which are defined as:

$$R1 = \Omega_0 = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$M3 = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$M4 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$M7 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- in $\mathcal{C}_{\mathcal{P}}$ we find R2/Kl (Klein's model), R5 (inverse paralogy [PR18]), M5, M6, and Ω, which are defined as:

$$R2 = Kl = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$R5 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$M5 = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$M6 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\Omega = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- in $\mathcal{C}_{\{Rev,IR,SymC\}}$ we find R3 (reverse analogy [PR18]), which is defined as:

$$R3 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

- in $\mathcal{C}_{\{IR,Rev,SymC,Sym::Sym\}}$ we find R4 (paralogy [PR18]), which is defined as:

$$R4 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

- in $\mathcal{C}_{\{IR,Sym::Sym\}}$ we find R6, which is defined as:

$$R6 = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- in $\mathcal{C}_{\{IR\}}$ we find R7 and R8, which are defined as:

$$R7 = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$R8 = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Models in $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ and $\mathcal{C}_{\mathcal{P}}$}. Figure C.2 illustrates how the Boolean models considered in [PR18] are separated in two groups, one with supersets of $Kl = \Omega_0 \cup S_2$ (with $S_2 = \{0110, 1001\}$, notation from [PR18]) and the other with the other supersets of Ω_0. These two groups differ mainly by whether S_2 is accepted or not. Furthermore, S_2 appears in all Ω, Kl, M_5, M_6, all models satisfying Symmetry of Ratio postulate (corresponding to $\mathcal{C}_{\mathcal{P}}$). Conversely, S_2 does not appear in Ω_0, M_3, M_4, M_7, all models that do not satisfy Symmetry of Ratio (corresponding to $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$).

To reformulate, $\{0101, 1010\}$ is present in all Boolean models mentioned by Prade and Richard, and it generates $S_2 = \{0110, 1001\}$ by application of the Symmetry of Ratio postulate ($Sym :: Sym$). Having $\{0101, 1010\}$ without S_2, as is the case with Ω_0, M_3, M_4, M_7, results in the model not being stable by application of $Sym :: Sym$, and thus it does not belong to $\top$.

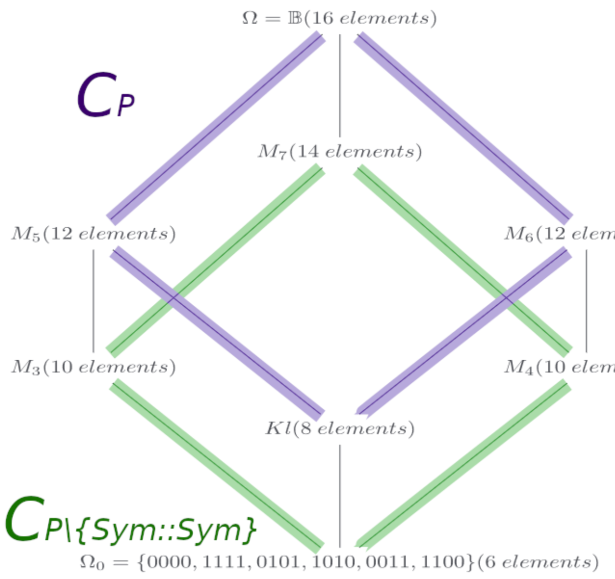

$\Omega = \mathbb{B}(16\ elements)$
$\mathcal{C}_P$
$M_7(14\ elements)$
$M_5(12\ elements)$
$M_6(12\ elements)$
$M_3(10\ elements)$
$M_4(10\ elements)$
$Kl(8\ elements)$
$\mathcal{C}_{P\setminus\{Sym::Sym\}}$
$\Omega_0 = \{0000, 1111, 0101, 1010, 0011, 1100\}(6\ elements)$

Figure C.2: Lattice of Boolean models of analogy according to Prade and Richard [PR18, Figure 1]. We highlight models that fit all the postulates, *i.e.*, $\mathcal{C}_{\mathcal{P}}$ (purple) and models that fit only in $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ (green), *i.e.*, $\mathcal{C}_{\mathcal{P}}$ without Symmetry of Ratio.

Differentiating models of the same equivalance class using seeds. As we can see from the above case, and as was extensively studied by Prade and Richard [PR18], different Boolean models can follow the same postulates of analogy (the ones of APs in [PR18]) but remain distinct, as correspond to different visions of analogy in the Boolean space. However, as illustrated in Figure C.2, the permutations postulates are not sufficient to distinguish all the models on $\{0, 1\}$. The existence postulates in Table C.1 fill part of this gap: by applying Identity to $\{0, 1\}$, we directly obtain $\{0000, 1111, 0011, 1100\}$, and by applying Reflexivity of Conformity we obtain $\{0000, 1111, 0101, 1010\}$. Applying all the postulates of $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ on either of the above results in Ω_0, and applying Symmetry of Ratio results in Kl. However, the other models are not generated with this process: 0111 and its permutations by $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$, but are necessary to obtain M_3, and, if $\mathcal{C}_{\mathcal{P}}$ is used instead of $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$, to obtain M_5. Similarly, 0001 and its permutations by $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ are necessary to obtain M_4, and M_6 by $\mathcal{C}_{\mathcal{P}}$. If both $0111, 0001$ are used in $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$, M_7 is obtained, and Ω is obtained from $0111, 0001$ and $\mathcal{C}_{\mathcal{P}}$.

There exist multiple justifications for these seeds, depending on which postulate are considered. While the postulates in Table C.1 are not enough to describe all the models, other postulates have been proposed by Leemhuis and Özçep [LÖ23], in particular the existence postulates ratio symmetry, universal neutrality, k-neutrality, and universality.

C.3 Postulates and data augmentation for ML

The data augmentation procedure proposed by Lim, Prade, and Richard [LPR19; LPR21] was used to train DL models in [LPR19; LPR21] and subsequent work, including ours [Als+21a; Als+21c; Mar+22a; MC24]. It can be separated into two parts: the augmentation of the number of valid APs, and the generation of invalid APs. This process is discussed in Appendix C.3.1.

In our work [MMC22], other equivalence classes are explored beyond the usual postulates of analogical proportions, that correspond to not accepting Exchange of the Means or considering it an undesirable property for quadruples. In particular, we use $\mathcal{C}_{\{Rev,IR,SymC\}}$ with use different ways to obtain non-analogies. In the same line of thought, we propose to extend the data augmentation procedure for valid and invalid APs to any subset of postulates, based on the categories mentioned at the beginning of Appendix C.2.1.

We saw in Appendix C.2.2 that any given set of postulates results in an equivalence class of permutation, and the data augmentation for valid APs can easily be generalized to any set of analogical quadruples using this notion. For APs, for each valid AP $A : B :: C : D$ in the dataset, we apply all the permutations of the equivalence $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ on $A : B :: C : D$ to obtain the 8 equivalent forms. Replacing $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ by any equivalence class induced by some postulates generalizes the procedure to any set of postulates.

However, more consideration must be given to the generalization of the generation of invalid APs. In Appendix C.3.2, we propose and discuss multiple methods to obtain and increase the amount of non-analogical quadruples.

Finally, we propose our generalized analogical data augmentation procedure in Appendix C.3.3.

C.3.1 Reminder of the principles underlying the data augmentation process

In the work of Lim, Prade, and Richard [LPR19; LPR21] and subsequent work, including ours [Als+21a; Als+21c; Mar+22a; MC24], data augmentation was used to train ML models (DL models, to be specific) for analogy detection.

To achieve this, two sets of training examples are required: examples of analogies, and examples of non-analogies. To obtain these, Lim, Prade, and Richard [LPR19; LPR21] proposed to use the permutations from what we identify as $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ to augment analogical exemplars with the 8 equivalent forms from Lepage [Lep01], and use the following generation rules to generate exemplars out of the $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$:

$$A : B :: C : D \implies B : A \not:: C : D, \tag{NA_1}$$
$$A : B :: C : D \implies C : B \not:: A : D, \tag{NA_2}$$
$$A : B :: C : D \implies A : A \not:: C : D. \tag{NA_3}$$

More specifically, applying (NA_1), (NA_2) on all of $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ generates all permutations of 4 distinct elements out of $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$ generated from $A : B :: C : D$. Additionally, the role of (NA_3) relates to Strong Identity and by extension Strong Reflexivity of Conformity, that become invalid after applying (NA_3) on all of non-analogies.

C.3.2 Non-analogies from postulates

In reference to the work Lim, Prade, and Richard [LPR19; LPR21], we separate non-analogies generated from a base of analogical exemplars in 5 groups:

1. permutations (without repetitions) of 4 elements in the equivalence class of a non-analogical exemplar;
2. permutations (without repetitions) of 4 elements falling out of the equivalence class;
3. permutations (with repetitions) of 4 elements conflicting with a constraint postulate;
4. permutations (with repetitions) of 4 elements conflicting with an existence postulate;
5. random combinations of elements from multiple exemplars, which assume that analogies are a minority in the data domain, thus random quadruples are likely to be non-analogical.

Case 1: permutations without repetitions of non-analogies. The principle used for analogical quadruples can be applied to non-analogical quadruple. Indeed, if at least one quadruple is analogical in the class, by equivalence all the others are valid. Conversely, if one quadruple is non-analogical then no other quadruple in the class is analogical. If we consider the law of excluded middle[3] for analogical quadruples, defined in Definition 3.1, then the equivalence class of a non-analogical quadruple contains only non-analogical quadruples.

Definition 3.1: Postulate $\mathcal{A}_1$: law of excluded middle on analogical statements.

A quadruple is either analogical or non-analogical.

This principle can be applied on non-analogical data present in the dataset, or on non-analogical data generated from analogical data using the principles listed below.

Case 2: permutations without repetitions, out of the equivalence class. The permutation postulates state that given an analogical quadruples, other quadruples are analogical, but they do not state that any other permutation is non-analogical. Therefore, to use exemplars from Item 2 as non-analogies, as done with $(NA_1), (NA_2)$, it is necessary to assume that permutations not explicitly analogical are non-analogical, which is an expression of the CWA[4] on the equivalence classes.

Definition 3.2: Postulate $\mathcal{A}_2$: CWA on equivalence classes.

Any quadruple that is a permutation of an analogical exemplar but not in the equivalence class of any analogical exemplar is non-analogical.

Let us use an example with equivalence $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$: we are able to partition the set of all 24 permutations of 4 elements A, B, C, D in three equivalence classes following $\mathcal{C}_{\mathcal{P}\setminus\{Sym::Sym\}}$, respectively containing $A : B :: C : D$, $B : A :: C : D$, and $C : B :: A : D$ [Lep03, Lemma 2 page 119]. From there, stating that one of the 3 equivalence classes is made of analogical exemplars does not inform us about the analogical nature of quadruples in the other 2 classes. However, by stating the CWA on the equivalence classes, having only members of the class of $A : B :: C : D$ as analogical exemplar allows us to determine $B : A :: C : D$, $C : B :: A : D$, and their equivalence classes as non-analogical. This principle was used in $(NA_1), (NA_2)$ to generate non-analogies [Als+21c; LPR19; LPR21; Mar+22a; MC24, and related articles].

The above reasoning can be extended to all equivalence classes from Appendix C.2.2, but we distinguish two variants:

- the greedy variant: given an analogical exemplar $A : B :: C : D$, **all** quadruples in the other equivalence classes are seen as non-analogical (version used by Lim, Prade, and Richard [LPR19; LPR21]):
- the tolerant variant: same as the above, but allows for multiple related analogical equivalence classes if at least one permutation from each of them is an analogical exemplar (version used by us in our experiments from Section 7.6 [Mar+22a]).

Case 3: conflicts with a constraint postulate. This kind of non-analogical quadruple is pretty self explanatory, for some accepted constraint postulate: if a quadruple conflicts with the constraint postulate, it cannot be analogical, otherwise the constraint postulate would not hold. If, similarly to case 1, we assume the law of excluded middle (see Definition 3.1), then quadruple conflicts with the constraint postulate are non-analogical.

For instance, for any three elements $A, B, B', B \neq B'$, the quadruple $\langle A, B, A, B'\rangle$ is in direct conflict with the Strong Reflexivity of Conformity postulate and $\langle A, B, A, B'\rangle$ is in conflict with the Strong Identity postulate. Taking A, B from an analogical exemplar and B' from another

[3]The law of excluded middle, in logic, states that a statement is either true or false, with no other possible value. Therefore, if a statement is not true, it must be false, and conversely.

[4]The CWA, as aptly defined in Wikipedia, is "the presumption that a statement that is true is also known to be true. Therefore, conversely, what is not currently known to be true, is false. [. . .] The opposite of the closed-world assumption is the OWA, stating that lack of knowledge does not imply falsity."

exemplar can ensure that A, B, B' are sound data point. The same principle can be applied to any other constraint postulate.

Another example of case 3 is the one of the Uniqueness postulate. Given the analogy $A : B :: C : D$, the quadruple $\langle A, B, C, D' \rangle, D' \neq D$ is in direct conflict with the Uniqueness postulate. Thus, if we assume Uniqueness, generating a quadruple $\langle A, B, C, D' \rangle$ for any $D' \neq D$ will result in a non-analogy. Taking A, B, C from an analogical exemplar and D' from another exemplar can ensure that A, B, C, D' are sound data points.

Case 4: conflicts with existence postulate. In some cases and depending on the formulation, it may be possible to generate triples conflicting with existence postulate, for instance Solvability. It appears harder to create conflicts with other existence postulate. However, the generation process is domain dependent, and will probably need to rely on heuristics.

One such example is what we propose in Chapter 11, that relies on a form of CWA on annotations: if a SR label A is present in two sentences s, t, and a label B is present in sentence s but not in t, from the CWA on annotations no label B can be found in t and thus $A_s : B_s :: A_t : x$ is non-analogical. We assume the CWA on annotations as the data is human annotated, and some labels may be forgotten. The use of the local context, here the sentences s, t and their annotations, may serve as an inspiration for heuristics in other domains.

Note that another assumption needs to be made to truly consider $A_s : B_s :: A_t : x$ as non-analogical, namely, that the absence of label is not a valid element (otherwise the solution would be $x = \emptyset$). We did not make in Chapter 11, and we considered that properly identifying the lack of elements to label was a success in terms of analogy solving. Based on these experiments, we consider that these unsolvable analogical equations constitute a class of exemplar distinct from analogical and non-analogical exemplars. It is however possible to leverage them by using any arbitrary element in place of the solution; the resulting quadruple will always be non-analogical, as the original triplet is unsolvable.

Case 5: random quadruples. The Item 5 group can be seen as closely related to the CWA: a plausible reformulation of the CWA would be that any quadruple that is not an analogical exemplar and cannot be obtained from chosen permutation or existence postulates is non-analogical, as stated in Definition 3.3. Therefore, any random quadruple of 4 distinct elements can be seen as non-analogical in the CWA. Note that Definition 3.3 is a stronger assumption than Definition 3.2.

Definition 3.3: Postulate $\mathcal{A}_3$: CWA on analogical exemplars.

Any quadruple that is not in the equivalence class of any analogical exemplar is non-analogical.

A relaxation of this principle is to consider the CWA only for elements involved in or non-analogical exemplars, *i.e.*, a local version of the CWA.

C.3.3 Generalized data augmentation algorithm

In this subsection we propose a set of algorithms and show how to combine them to apply data augmentation based on axiomatic setting. The proposed formulation of the algorithms is intended to make obvious the assumptions made at each step of the process, and to be flexible and controllable. Indeed, we can adapt the process by deciding which postulates, assumptions on the data, and heuristics for non-solvable analogical equations we use.

As a reminder, we mainly consider assumptions $\mathcal{A}_1$[5], $\mathcal{A}_2$[6], and $\mathcal{A}_3$[7].

Summarized generalized data augmentation procedure. The procedure can be summarized as follows:

1. augment analogical exemplars:
    - with the equivalence class of the chosen permutation postulates;

[5]CWA on the equivalence classes: any non-explicitly analogical equivalence class is non-analogical.
[6]Law of excluded middle on analogical statements: any quadruple is either non-analogical or analogical.
[7]CWA on analogical statements: Any quadruple that is not in any equivalence class of an analogical exemplar is non-analogical.

Algorithm 2: Applying a permutation or a permutation postulate.

```
Permutations postulates are expressed using the initial position of each element in the
  quadruple (starting from 0), reordered following the postulate. For instance, $(1, 0, 3, 2)$
  corresponds to $A : B :: C : D \implies B : A :: D : C$.

Input: A permutation postulate $a \in [0, 3]^4$, a quadruple $p = (A, B, C, D)$ to permute
Output: $p'$, the result of applying $a$ on $p$.
With $a_i$ the $i$-th element of $a$. With $p_i$ the $i$-th element of $p$.
We reorder $p$ following $a$:
$p' \leftarrow (p_{a_0}, p_{a_1}, p_{a_2} p_{a_3})$
```

- with the existence postulates, if needed;

2. generate non-analogical exemplars using either:
   - permutations of analogical exemplars out of the equivalence class, under $\mathcal{A}_1$;
   - quadruples conflicting with a constraint postulate, under $\mathcal{A}_2$;
   - quadruples conflicting with an existence postulate, using task dependant heuristics;
   - random combinations of elements from multiple exemplars, under $\mathcal{A}_3$;
3. augment non-analogical exemplars with the equivalence class of the chosen permutation postulates, under $\mathcal{A}_2$.

The order of the steps is designed to minimize necessary computations and ensures that all possible non-analogical exemplars are generated.

Balancing generalized data augmentation procedure. For training, we recommend to apply a sampling algorithm to balance analogical and non-analogical exemplars. Ideally, balancing should also be performed between sources of non-analogical exemplars, to maximize diversity. This can be done by limiting the number of quadruples generated at each step above, by applying the steps above on the original data until a certain amount of each type of non-analogical exemplar is obtained. It is also possible to do the selection a posteriori.

Compared to this macro scale balancing, it is also possible to apply limits at the micro scale and ensure each analogical exemplar is represented equally. For instance, inSubsections 6.3.3 and 7.3.3 [Mar+22a], we sample 8 out of the 24 available non-analogical quadruples generated from each analogical exemplar of the original data, to match the 8 analogical quadruples generated from the equivalence class.

Merging steps and representig permutations. To simplify manipulation, we express the permutations using only the initial position of each element in the quadruple. For instance, for a base quadruple A, B, C, D, $(0, 1, 2, 3)$ corresponds to A, B, C, D itself, and $(1, 0, 3, 2)$ to B, A, D, C. We use a similar principle to represent the permutation postulates: $(1, 0, 3, 2)$ corresponds to $A : B :: C : D \implies B : A :: D : C$, as by applying the postulate we reorder the original quadruple following $(1, 0, 3, 2)$. Following this representation, Algorithm 2 is an example of how to apply some permutation postulates on a quadruple, and we use Algorithm 3 to generate the equivalence classes. One advantage of this encoding is that many of the steps above can be computed on the permutations directly, as a way to compute permutations beforehand and avoid the potentially costly direct manipulation of the elements in the quadruples. Once the permutation classes have been computed, they can be directly applied to the manipulated object.

In particular, some of the possibilities in step 2. as well as step 3. of the procedure can be merged with step 1. in the implementation. In particular if the set of postulates is not going to change through the process, many of the values can be pre-computed. The data augmentation used by Lim, Prade, and Richard and in our work [LPR19; LPR21; Mar+22a; MMC22] are examples of that.

Algorithm 3: Equivalence class generation.

```
Permutations are expressed using the initial position of each element in the quadruple. For
  instance, for a base quadruple $A : B :: C : D$, $(0, 1, 2, 3)$ corresponds to $A : B :: C : D$
  itself, and $(1, 0, 3, 2)$ to $B : A :: D : C$.

Input: A set of permutation postulates $A$
Output: The equivalence class of permutations $P$, expressed using the initial position of
        each element in the quadruple.
/* We initialize the equivalence class of permutations with the initial
   position of each element.                                                    */
$P \leftarrow \{(0, 1, 2, 3)\}$;

/* We repeatedly apply all the permutation postulates untill the
   equivalence class is stable by application of said postulates.               */
$stable \leftarrow False$;
while $i < 4^4$ and $\neg stable$ do
    $stable \leftarrow True$;
    $i \leftarrow i + 1$;
    foreach $a \in A$ do
        $P' \leftarrow \{\forall p \in P,$ apply $a$ on $p\}$;

        /* If, using any of the postulates, we generate a permutation not
           already contained in $P$, the equivalence class is not stable yet.
           */
        if $|P' \setminus P| > 0$ then $stable \leftarrow False$;

        $P \leftarrow P \cup P'$;
    end
end
```

C.4 Some recommendation and guidelines to adapt the framework of axiomatic analogy to specific use-cases

In this appendix, we discuss the impact of some aspects of a task tackled by analogy detection or analogy solving on the generalized data augmentation procedure and the formulation of the problem. This section has no normative purposes, and simply reflects our experience and thought on applying analogical reasoning on a variety of settings, and in particular with axiomatic APs.

From our experiments in Chapters 10 and 11, we found the analogical formulation of a task, combined with a suitable analogical model, bring several benefits. For instance, using exemplar-based processes, the models can achieve high performance with simpler structures and fewer parameters, as the similar problems (for instance, labelling the SRs for different semantic frames) use the same model parameters and formulation. In Chapters 10 and 11 and Section 7.5, the models that were also tolerant to variations and perturbation in the input, and more stable across random initialization of the parameters than comparable non-analogical models. This last observation might be caused by the variety of training examples obtained using analogical data augmentation.

The input and expected output of a task can significantly impact how the task is formulated. Below we list some major types of applications and how we would proceed to model them.

Note that analogy detection, analogy solving, and analogical inference are all meaningful to learn representations, or to learn relations. Additionally, we only describe tasks for which ve have some kind of experience, and the guidelines below should be extended in the future to cover as many types of ML tasks as possible.

Prediction tasks using analogy solving. Many regression and classification tasks can be seen as datum prediction tasks: given an input X, we want to predict a label or output $Y = f(X)$. We separate these tasks into datum prediction and datum completion. We call datum prediction tasks any task where there is a need to predict an unknown datum D, possibly under some constraints.

We call datum completion tasks a subset of datum prediction tasks, where part of the datum X are known, written X_k, and we want to predict the unknown parts X_{uk}.

The two tasks can be formulated as analogy solving, but such a formulation requires a reference datum C and a reference ratio $A : B$, that we want to report on C to obtain X. The resulting analogical equation is $A : B :: C : x$, solved by $x = X$ For datum completion, as we already know X_k, the search space can be restricted to values x which have X_k in their value ($x_k = X_k$). A literal interpretations of analogy solving for regression tasks can be found in most of the approaches we identified as using analogy solving on APs, for instance for image retrieval or generation [Bay+07; Bit+23; LTC17; Lep14; Ree+15; SZF15] given three reference images A, B, C. In these applications, the task is directly formulated as analogy solving.

What is more interesting, in our opinion, is that any prediction task can be reformulated as analogy solving. Given an input X, to predict a label or output $Y = f(X)$, we can find reference pairs (X', Y') in the training set that are such that $Y' = f(X')$, and solve the analogical equation $X' : Y' :: X : x$. This is what we did in Chapter 11 to predict the group of words that have a given SR label for the FSRL task. One advantage of this formulation is that the same model can be used for multiple similar prediction tasks without changing the formulation, and using the same model, which benefits from the transfer of knowledge between tasks. In such a setting, it might be interesting to not consider Inversion of Ratio if f is not invertible

Prediction tasks using analogical inference. Another formulation of datum prediction tasks is through analogical inference: if the function f is analogy preserving [Cou+17b], then it is possible to use the analogical inference principle discussed in Subsection 2.2.1. This is the underlying principle behind prediction using CoAT [Bad20].

In the case of datum completion, assuming the link between X_k and X_{uk} is analogy preserving, analogical inference can also be used. In such a setting, choosing the postulates such that the corresponding model of analogy is preserved could allow to apply the inference principle on functions that are not strictly analogy preserving in the terms of Couceiro, Hug, et al. [Cou+17b].

Prediction tasks using analogy detection. Analogy detection is particularly suited to relation checking tasks where we want to know if two elements C, D share the same relation as a pair A, B. This principle was used in the work of Jarnac, Couceiro, and Monnin [JCM23] to determine if a knowledge graph node n_2 in the graph generated from a node n_1 should be pruned or not, based on past decision represented by pairs (n'_1, n'_2). From there, if $n'_1 : n'_2 :: n_1 : n_2$ holds, then the same decision holds for n'_2. Another example is the work of Alsaidi, Couceiro, et al. [Als+22b] where ordinal relations are considered, in particular the precedence of a medical record over another medical record.

This principle can be extended for any classification task based on some contextual information, by considering analogies where $ctx_1 : obj_1 :: ctx_2 : obj_2$ holds if the class label of obj_1 in context ctx_1 is the same as the one of obj_2 in context ctx_2. In the work of Jarnac, Couceiro, and Monnin, the context was specified by the seed nodes n_1, n'_1 used to generate the graphs of n_2, n'_2 respectively. Using this process for contextualized classification implies that Uniqueness does not hold, as multiple objects can have the same class label in the same context. As an example, for the node pruning task in Jarnac, Couceiro, and Monnin, multiple nodes from the same graph will have the "has to be pruned" label, and multiple will have the "has to be kept" label.

There is another way to express classification tasks as analogy detection tasks, including classification tasks where the class is not explicitly available. For instance, it is possible to determine if C and D have the same class using a reference pair (A, B) that we know have the same class, by checking if $A : B :: C : D$ holds, as was done for instance in some of the experiments on medical records by Alsaidi, Couceiro, et al. [Als+22a]. In that case, Symmetry of Ratio is implied, as the ratio is an equivalence of labels.

Appendix D

Résumé étendu

D.1 Introduction

Les analogies sont un élément clé de la cognition humaine et peuvent être considérées comme un mécanisme d'abstraction qui identifie les similarités et différences entre différentes situations, et comme un outil de raisonnement permettant d'adapter des solutions connues à de nouvelles situations. Lorsque l'on raisonne par analogie, l'objectif est d'adapter la solution d'un problème connu ou source, qui est suffisamment similaire au problème réel ou cible. Ce processus implique un transfert entre le contexte du problème source (le problème et sa solution) et le contexte du problème cible. Ces dernières années ont été marquées par un regain d'intérêt pour le potentiel de la détection des analogies et de l'inférence analogique, avec des applications fructueuses dans le domaine de l'apprentissage automatique pour la détection des relations, la récupération et la génération d'images, de textes et d'unités de connaissances formelles telles que les graphes de connaissances. Si certains de ces travaux reposent sur une compréhension intuitive de l'analogie, des efforts considérables ont été déployés depuis l'Antiquité pour définir les analogies de la manière la plus précise possible. En particulier, une notion qui a suscité beaucoup d'intérêt au cours des 50 dernières années est celle de Proportion Analogique (PA). Une PA est typiquement composée de quatre éléments A, B, C, D, et signifie que la relation entre A et B est analogue à celle entre C et D, notée $A : B :: C : D$. Cet outil formel a été décliné pour couvrir de nombreuses interprétations différentes de ce que peut être une analogie, avec n'importe quel nombre d'éléments.

L'intérêt de la morphologie pour l'étude des PA. L'application du raisonnement analogique n'est pas aussi simple qu'il n'y paraît. La morphologie des mots est un point de référence très intéressant pour l'étude des PA, car il s'agit d'une forme d'analogie entre chaînes de caractères, qui peut généralement être généralisée à l'analogie entre chaînes de symboles. Applying analogical reasoning is not as easy as it seems. Des données morphologiques sont accessibles pour de nombreuses langues à l'ère de l'internet, et les transformations qui peuvent se produire par les mécanismes de l'inflexion morphologique ont été largement étudiées d'un point de vue linguistique. De plus, alors que l'inflexion morphologique est très régulière pour la plupart des cas (considérez la PA $dog : dogs :: cat : cats$ ajoutant le suffixe $-s$ à cat et dog), de petites variations et irrégularités sont présentes qui peuvent être difficiles à prédire sans connaissances linguistiques (considérez la PA $dog : dogs :: bus : buses$ où le suffixe $-s$ devient $-es$ avec bus). Dans un tel contexte, le raisonnement analogique peut être utilisé pour exploiter des exemples de transformations morphologiques et réaliser des prédictions explicables avec un minimum d'effort.

Par exemple, dans les années 1980, un outil nommé Copycat [HM95] a été proposé pour résoudre les PAs entre les chaînes de caractères, permettant, par exemple, de trouver la solution $x =$ "*pqs*" à l'équation analogique "*abc*" : "*abd*" :: "*pqr*" : x, ou $x =$ "*cats*" pour l'équation "*dog*" : "*dogs*" :: "*cat*" : x. D'autres approches symboliques ont depuis été développées pour traiter les PAs dans le domaine de la morphologie, et bien que leurs performances se rapprochent du comportement humain, l'aspect irrégulier de la morphologie des mots est une limitation commune.

Le framework ANN pour manipuler les PAs morphologiques en utilisant le DL. Dans ce contexte, nous adaptons et développons une approche récente d'apprentissage profond (deep learning, DL) basée sur les PAs [LPR19] dont la performance surpasse celle de formulations plus rigides dans le domaine de la sémantique des mots. Nous proposons le framework ANN, qui comprend le réseau neuronal d'analogie pour la classification (Analogy Neural Network for classification, ANNc) et le réseau neuronal d'analogie pour la récupération/génération de solution (Analogy Neural Network for retrieval/generation, ANNr), deux modèles légers de DL pour aborder la détection et la résolution d'analogies, ainsi qu'un processus d'augmentation des données. Le framework est formellement ancré dans les travaux sur les PAs, et bénéficie de la flexibilité des modèles de DL pour traiter les irrégularités dans les données. Pour explorer les performances de notre approche, nous avons développé le jeu de données Siganalogies couvrant plus de 80 langues. Notre approche surpasse les approches symboliques par une marge significative sur la détection d'analogie, la résolution d'analogie par récupération, et la résolution d'analogie par génération, dans certains cas par plus de 75% d'exactitude. Nous réalisons des expériences d'ablation approfondies qui nous permettent d'affiner le framework ANN. Le fait de disposer de grandes quantités de données dans de nombreuses langues nous permet d'explorer les PAs morphologiques entre les langues, ce qui révèle des similitudes intéressantes entre notre modèle et la parenté diachronique des langues[1].

Extension du succès de le framework ANN à la sémantique et aux LLMs. Motivés par les succès de notre framework ANN sur les PAs morphologiques, nous adaptons le cadre aux domaines d'application au-delà de la morphologie des mots. Par exemple, nous développons un modèle basé sur BERT et le framework ANN pour résoudre TSV, et surpasser l'état de l'art sur le benchmark WiC-TSV. Nous proposons également un système tirant parti de PAs sémantiques basé sur la sémantique des cadres (Frame Semantics) pour s'attaquer à l'annotation de rôle sémantique (Frame Semantic Role Labeling, FSRL), sur le jeu de données FN1.7. Cette nouvelle méthode se révèle flexible et, bien que certains défis subsistent dans la sélection de la source pour le transfert analogique des annotations de rôle sémantique, elle a le potentiel de surpasser les méthodes état de l'art.

Proportions analogiques au delà de le framework ANN. Enfin, nous explorons le potentiel d'une mesure de complexité pour le transfer analogique (Complexity Measure for Analogical Transfer, CoAT) pour plusieurs aspects de la maintenance de base de cas (, case base), un problème important dans le domaine du raisonnement à partir de cas (Case-Based Reasoning, CBR). Les approches de CBR considèrent un ensemble de cas, qui décrivent des situations et leurs issues, ou des problèmes et leurs solutions, et s'appuient sur cette case base pour résoudre de nouveaux problèmes ou prédire l'issue de nouvelles situations. Dans ce contexte, nous avons pu définir une mesure de la compétence, ou de l'utilité, des cas dans une case base, ce qui nous a permis d'améliorer la qualité de case bases pour résoudre des tâches de classification sur des données synthétiques et réelles.

Structure du document et de ce résumé. Cette thèse est séparée en trois parties. Tout d'abord, la Partie I contient les chapitres introductoires aux notions et approches clés mentionnées dans cette thèse, à savoir l'analogie (Chapitre 2), DL (Chapitre 3), et la morphologie (Chapitre 4). Ensuite, la Partie II décrit nos contributions sur le framework ANN dans le cadre morphologique, tandis que la Partie III regroupe nos contributions à TSV, FSRL, et au CBR. Les trois parties sont précédée par un chapitre introductoire, le Chapitre 1, repris dans la présente section.

Dans ce résumé étendu, nous détaillons les principales contributions et conclusions des Parties II et III : le framework ANN est détaillé en Appendix D.3, accompagné de nos conclusions sur les données morphologiques ; les applications à TSV et FSRL sont résumées dans la Appendix D.4 ; enfin, les contributions à l'approche CoAT sont expliquées dans la Appendix D.4.3. Nous détaillons aussi les points principaux abordés dans la Partie I.

D.2 Partie I : préliminaires et état de l'art

La première partie de ce manuscrit est consacrée à l'ensemble des notions nécessaires pour comprendre les contributions présentées.

[1]La parenté diachronique des langues compare leur évolution au cours de l'histoire.

Postulat	Symbole	Description
		Groupe des postulats de permutation
Symétrie de la Conformité	$SymC$	$A : B :: C : D \implies C : D :: A : B$
Échange des Moyens	EM	$A : B :: C : D \implies A : C :: B : D$
Inversion des Rapports	IR	$A : B :: C : D \implies B : A :: D : C$
Échange des Extrêmes	EE	$A : B :: C : D \implies D : B :: C : A$
Symétrie de lecture	Rev	$A : B :: C : D \implies D : C :: B : A$
Postulat supplémentaire 1	$Ex1$	$A : B :: C : D \implies C : A :: D : B$
Postulat supplémentaire 2	$Ex2$	$A : B :: C : D \implies B : D :: A : C$
*Symétries des Rapports	$Sym :: Sym$	
**Symétrie du Rapport gauche*	$Sym ::$	$A : B :: C : D \implies B : A :: C : D$
**Symétrie du Rapport droit*	$:: Sym$	$A : B :: C : D \implies A : B :: D : C$
		Groupe des postulats d'existance
Réflexivité de la Conformité	Ref	$A : B :: A : B$
Identité	Id	$A : A :: B : B$
Solvabilité	$Solv$	$\forall A, B, C \quad \exists D$ tel que $A : B :: C : D$ soit une PA
		Groupe des postulat de contrainte
Unicité	$Uniq$	$A : B :: C : D \wedge A : B :: C : D' \implies D = D'$
Réflexivité Forte de la Conformité	$UniqRef$	$A : B :: A : D \implies B = D$, peut être vue comme l'Unicité restreinte à la Réflexivité de la Conformité
Identité Forte	$UniqId$	$A : A :: C : D \implies C = D$, peut être vue comme l'Unicité restreinte à l'Identité
Distribution	$Dist$	$A : B :: C : D \implies \mathcal{X}(A) \subseteq \mathcal{X}(B) \cup \mathcal{X}(C)$, avec $\mathcal{X}(A), \mathcal{X}(B), \mathcal{X}(C)$ les propriétés de A, B, C

Table D.1: Postulats connus utilisés pour les analogies axiomatiques, excluant la transitivité. Les astérisques indiquent les postulats de permutation qui ne sont pas acceptés pour les proportions analogiques.

D.2.1 Chapitre 2 : analogie et proportions analogiques

Dans le Chapitre 2 nous introduisons les subtilités de la notion d'analogie et décrivons la notion plus précise de proportion analogique (PA) et d'équation analogique.

En particulier, en fonction du contexte, l'analogie peut décrire des notions distinctes mais apparentées. Nous tentons de donner un aperçu de ces notions et de la manière dont elles sont liées dans la Section 2.1, en nous appuyant sur une catégorisation de l'analogie dans la langue naturelle proposées par Barbot, Miclet, and Prade [BMP19, Section 2.1]. Dans nos travaux, nous nous concentrons sur la notion de PA. Telle que définie dans la Subsection 2.1.1, une PA est une relation quaternaire entre des éléments généralement de même nature, écrite $A : B :: C : D$ et se lisant "A est à B comme C est à D". Une PA où un des élément est une inconue x, noté $A : B :: C : x$, est appelé une équation analogique. Pour illustrer la polyvalence de APs, nous énumérons quelques applications intéressantes dans la Section 2.2.

Dans la Section 2.3, nous présentons le cadre formel que nous utilisons pour les approches développées dans cette thèse. Ce cadre formel suit les travaux fondateurs de Lepage and Ando [LA96] en exploitant les postulats des PAs, résumés dans la Table D.1.

Nous introduisons certaines limitations du cadre formel dans la Subsection 2.3.2, qui sont examinées plus en détail dans la Subsection 6.3.2, la Section 7.6, et l'Appendix D. Pour donner au lecteur une vue d'ensemble des formulations possibles du problème de l'analogie, nous présentons également une brève introduction à d'autres cadres formels clés dans la Section 2.4.

D.2.2 Chapitre 3 : apprentissage automatique, apprentissage profond, et représentations vectorielles

Le Chapitre 3 est consacré aux concepts de base liés à l'apprentissage profond (DL), qui sont nécessaires à la compréhension de cette thèse.

Le DL est une branche de l'apprentissage automatique (ML), dont l'objectif est de définir et d'optimiser un modèle paramétrique pour une tâche spécifique. Ces notions sont expliquées dans la Section 3.1. Habituellement, les paramètres les plus appropriés sont ceux qui permettent d'obtenir les performances les plus élevées pour un modèle donné, mais d'autres aspects sont pris en compte, tels que la capacité de généralisation à des données inédites et plusieurs notions d'équité et d'explicabilité.

En DL, un modèle suit une structure appelée l'architecture du modèle, qui est généralement composée de blocs de construction préexistants. Les blocs de construction mentionnés dans cette thèse sont décrits dans la Section 3.2 : le neurone artificiel, le Perceptron et Perceptron Multi-Couche, le réseau de neurone convolutif (Convolutional Neural Network, CNN), ansi que les modèles usuels pour traiter des entrées séquentielles comme le texte (LSTM et transformer). Une partie importante de l'exécution des tâches à l'aide de DL consiste à trouver la meilleure représentation des objets manipulés par le modèle. Dans Section 3.3, nous décrivons les principales méthodes permettant d'obtenir un tel vecteur de représentation, appelée embedding, y compris les notions d'auto-encodeur (AE), de pre-entrainement, les architectures Word2Vec, GloVe et FastText utilisées dans la Partie II et l'architecture BERT utilisées dans les Chapitres 10 et 11.

Une fois ces architectures et principes fondamentaux expliqués, nous examinons dans Section 3.4 certaines applications des PAs dans le domaine du DL, et en particulier les PAs qui manipulent les embeddings. Dans cette section, nous introduisons la règle du parallélogramme pour quatre embeddings $\vec{A}, \vec{B}, \vec{C}, \vec{D}$, qui peut s'écrire par exemple :

$$\begin{aligned}
\vec{A} - \vec{B} &= \vec{C} - \vec{D} \\
\vec{A} + \vec{D} &= \vec{B} + \vec{C} \\
\vec{D} &= \vec{C} - (\vec{A} - \vec{B}) \\
\vec{D} &= \vec{C} + \vec{B} - \vec{A}.
\end{aligned}$$

Nous décrivons également deux méthodes, 3CosAdd [Mik+13] et 3CosMul [LG14], utilisées pour résoudre des équations analogiques pour des modèles d'embedding modernes tels que Word2Vec, GloVe et FastText. Ces approches sont limitées par le manque de flexibilité inherent à des formules définies, ce qui résulte en des performances significativement différentes de celles d'un humain [CPG17; RDL17]. Pour remédier à cette limitation, plusieurs approches ont été proposées pour tirer parti d'exemples de PA pour entraîner des modèles ou des embeddings adaptés à la manipulation d'analogies. En particulier, dans nos travaux nous nous basons sur l'approche de Lim, Prade, and Richard [LPR19], qui à proposé d'apprendre un réseau neuronal à deux couches pour la détéction et la résolution d'analogie sur des embeddings. Les modèles en questions sont décrits dans la Section 6.2 et ci dessous.

D.2.3 Chapitre 4 : la morphologie des mots

Le domaine d'application des approches et des expériences que nous présentons dans Part II est la morphologie. Nous expliquons dans le Chapitre 4 ce qu'est la morphologie, y compris une description de certaines approches clés.

Premièrement, nous présentons brièvement la morphologie et les transformations morphologiques dans Section 4.1. Nous donnons ensuite quelques exemples d'AP en morphologie et mettons en évidence deux liens importants entre l'analogie et la morphologie dans Section 4.2. Enfin, dans Section 4.3, nous présentons une variété d'approches informatiques de la morphologie et des PAs sur les relations morphologiques. A notre connaissance, à l'exception de l'approche que nous proposons dans la Partie II sur la base des travaux de Lim, Prade, and Richard [LPR19], toutes les approches des PAs morphologiques se concentrent sur la caractérisation formelle des PAs, avec un ensemble d'opérations conçues par des experts [LYZ09; Mur+20], des relations entre les caractères [HM95], ou des caractéristiques morphologiques [FL18].

Parmi les approches mentionnées dans ce Chapitre 4, nous comparons le framework ANN avec Alea [LYZ09], Kolmo [Mur+20], ainsi qu'un des outils proposés par Fam and Lepage [FL18] que nous appelons Nlg. Ces trois approches sont détaillées plus en profondeur dans le Chapitre 7 sur notre protocole experimental, plus spécifiquement dans la Section 7.2.

D.3 Partie II : ANN framework et analogies morphologiques

La Partie II décrit les principaux travaux réalisés sur le framework ANN. Pour ce faire, le Chapitre 5 présente le jeu de données Siganalogies utilisé tout au long de cette partie, ainsi que quelques exemples tirés de Siganalogies. Le framework même est décrit dans le Chapitre 6, et des résultats expérimentaux détaillés sont présentés dans les Chapitres 7 et 8. Enfin, dans le Chapitre 9, nous résumons les contributions présentées dans la Partie II et expliquons comment le framework ANN et Siganalogies sont mis à disposition selon les principes de la science ouverte, y compris les démonstrations interactives disponibles pour le grand public. Cette partie couvre les contributions publiées dans [Als+21a; Als+21b; Als+21c; Cha+22; Mar+22a; MC24; Mar+22b].

D.3.1 Chapitre 5 : jeu de données d'analogies morphologiques Siganalogies

Pour développer et évaluer les performances de le framework ANN sur les PAs morphologiques, nous avons conçu le jeu de données Siganalogies [Mar+22b]. Ce jeu de données contient des PAs morphologiques dans plus de 80 langues distinctes et est construit à partir de trois jeux de données : Sig16 [Cot+16], Sig19 [McC+19], et le JBATS [Kar+18]. Chaque jeu de données contient des mots liés par des transformations morphologiques, que nous utilisons pour créer APs comme expliqué dans Section 5.1. Nous présentons Sig16, Sig19 et JBATS dans la Section 5.2, et fournissons des statistiques détaillées sur Siganalogies dans la Section 5.3. Une description du jeu de données avec des statistiques détaillées est également disponible sur la page GitHub du jeu de données[2].

Nous concluons ce Chapitre par une discussion sur les limites de Siganalogies dans Subsection 5.5.3. Tout d'abord, il existe des différences de codage des caractères entre Sig16 et Sig19, expliquées dans Subsection 5.5.2. De plus, nous avons constaté des problèmes de représentativité dus à une distribution inégale et limitée des marqueurs de discours (Part Of Speech tags, POS) et des mots obsolètes, présentés dans Subsection 5.5.1. Le jeu de données Siganalogies a été développé et affiné tout au long de nos expériences, et de nombreuses perspectives d'amélioration et d'expériences supplémentaires subsistent. Nous les détaillons dans Subsection 5.5.3.

D.3.2 Chapitre 6 : le framework ANN

Dans ce chapitre, nous décrivons le framework ANN, illustré par Figure D.1. Le framework peut être divisé en trois éléments clés, à savoir le(s) modèle(s) d'embedding, les modèles d'analogie et l'augmentation des données.

Le framework ANN a été initialement proposé par Lim, Prade, and Richard [LPR21] pour résoudre des équations analogiques sémantiques, et a été conçu pour être utilisé avec des modèles d'embedding pré-entraînés. En particulier, une première approche plutôt naïve de l'augmentation des données pour l'entraînement analogique ainsi que deux modèles d'apprentissage profond ont été proposés dans [LPR21] pour s'attaquer à la détection des analogies et à la résolution des analogies.

Ces deux modèles, que nous appelons ANNc et ANNr, ont été affinés au fil de nos expériences. Nous avons également intégré plusieurs approches non paramétriques dans le framework ANN, telles que la règle du parallélogramme [Mik+13] et 3CosMul [LG14] (tous deux définis dans la Subsection 3.4.1). Tous ces modèles de manipulation d'analogie sur les espaces d'embedding sont décrits dans Section 6.2.

Le framework ANN est fondé sur l'analogie axiomatique : les architectures ANNc et ANNr sont basées sur des intuitions guidées par les postulats décrits de l'analogie axiomatique, et avec l'augmentation des données décrite dans Section 6.4, nous entraînons des modèles pour s'adapter à un ensemble donné de postulats en devenant invariants aux permutations correspondantes.

Il existe plusieurs différences majeures entre la version actuelle du framework ANN et la version de Lim, Prade, and Richard :

[2]https://github.com/EMarquer/siganalogies/blob/main/siganalogies_description.pdf

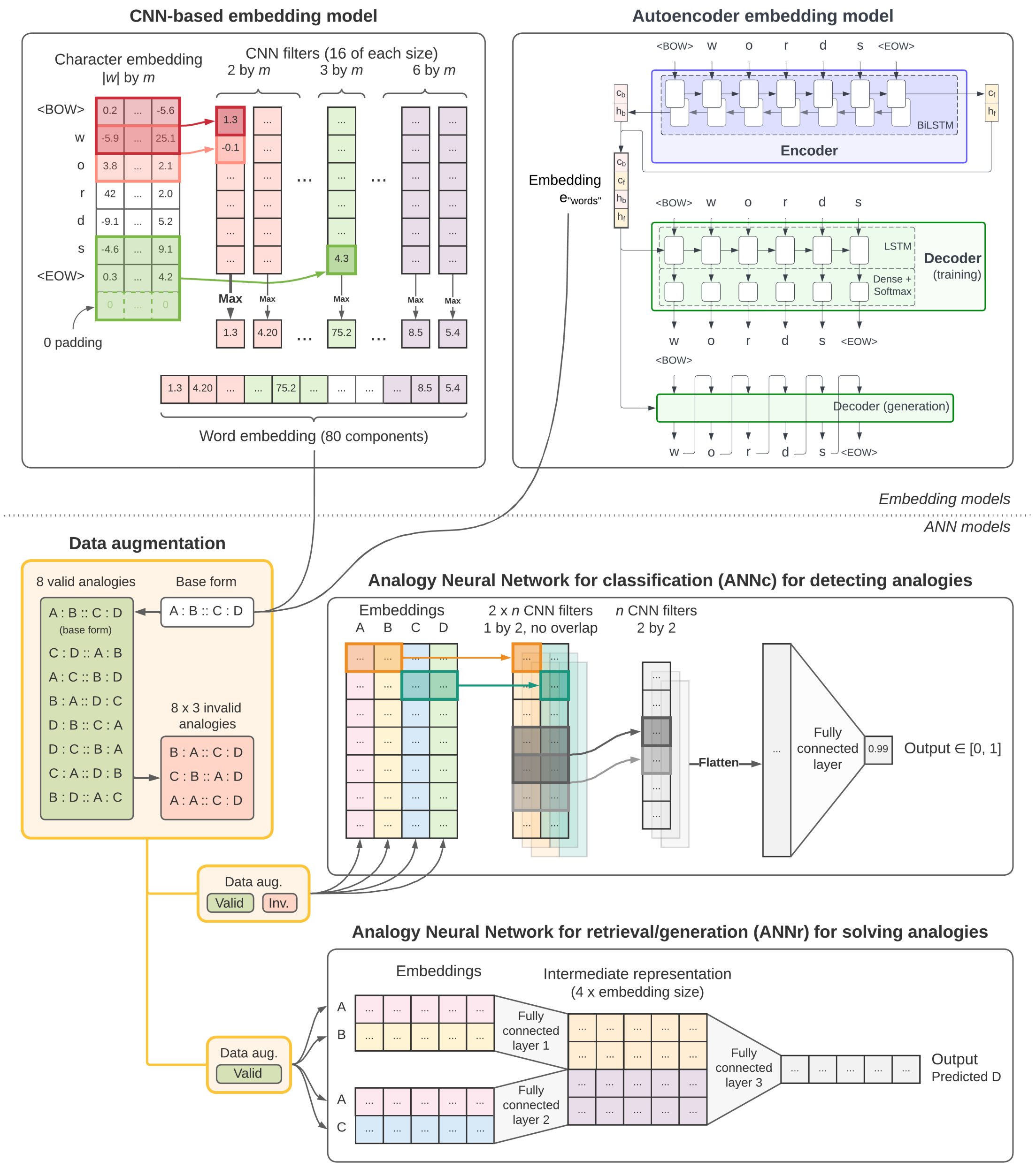


Figure D.1: Figure 1 de [MC24]. Légende d'origine: "Morphological embedding models, data augmentation, analogy classification (ANNc) and analogy retrieval (ANNr) models."

- pour obtenir des représentations appropriées des mots pour les PAs morphologiques, nous utilisons des modèles d'embedding sur mesure spécialisés pour la morphologie (voir Section 6.1) au lieu d'un modèle d'embedding pré-entraîné spécialisé pour la sémantique comme Lim, Prade, and Richard ;

- comme le modèle d'embedding n'est pas pré-entraîné et a été conçu pour avoir un petit nombre de paramètres, nous permettons un re-entraînement du modèle d'embedding pendant l'entraînement du modèle d'analogie (voir Section 6.4) ;

- pour atteindre de bonnes performances sur la résolution d'analogie avec ANNr lorsque des des modèles d'embedding non pré-entraînés sont employés, nous utilisons ANNc et la détection d'analogie comme tâche de pré-entraînement (voir Subsection 6.4.2) ;

- parce que nous permettons un re-entraînement du modèle d'embedding pendant l'entraînement avec ANNr, nous devons utiliser des objectifs d'entraînement plus raffinés que l'erreur quadratique moyenne (Mean Squared Error, MSE) (voir Subsection 6.4.2) ;

- nous améliorons le processus d'augmentation des données en équilibrant les PAs valides et invalides, et nous expérimentons avec des variations du cadre axiomatiques (voir Section 6.3).

Le framework ANN offre de multiples façons d'aborder la détection d'analogies et la résolution d'analogies. Malgré les alternatives que nous proposons, il est possible de considérer que l'architecture de ANNc et de ANNr n'est pas la plus adaptée à la manipulation de APs, bien qu'elle soit relativement facile à appréhender intuitivement et qu'elle permette d'obtenir de bonnes performances.

Nous détaillons dans le Chapitre 7 les expériences qui démontrent les performances du framework ANN sur la détection et la résolution d'équations analogique, et nous discutons des principaux avantages et inconvénients de chaque méthode dans les Sections 7.7.4 et 6.5. Dans le Chapitre 7, nous présentons également les résultats des expériences préliminaires qui ont guidé les décisions de conception mentionnées dans le Chapitre 6. Des discussions supplémentaires sur ces experiences sont présentes dans la Subsection 7.7.2.

D.3.3 Chapitre 7 : analyses quantitatives et qualitatives du framework ANN

Le Chapitre 7 détaille nos expériences avec le framework ANN sur le jeu de données Siganalogies. Dans la Section 7.7, nous concluons le chapitre par une discussion de nos contributions et des résultats présentés dans les Sections 7.3, 7.4, et 7.5.

Avant de décrire nos expériences, nous donnons quelques informations générales sur notre dispositif expérimental dans la Section 7.1, et nous résumons les aspects techniques modèles de référence dans la Section 7.2. Nous détaillons ensuite plusieurs expériences sur des modèles entraînés dans chaque langue pour la détection d'analogies dans Section 7.3, et la résolution d'analogies dans Section 7.4. Dans Section 7.5, nous réalisons une étude d'ablation sur les tâches de détection et de résolution d'analogies afin de déterminer leur tolérance au regard de perturbations de l'espace d'embedding.

Nous étudions également l'impact de l'augmentation des données analogique sur le comportement du modèle de détection des analogies lorsque la Permutation des Moyens est considérée différemment, suivant les discussions sur la manière de traiter les postulats des PAs qui ne conviennent pas à certains contextes d'application [Als+22b; Ant22, voir aussi Subsection 2.3.2]. Ces expériences sont décrites dans la Section 7.6, et se réfèrent à des variantes du processus d'augmentation des données défini dans Subsection 6.3.2.

D.3.4 Chapitre 8 : analyses quantitatives et qualitatives du framework ANN

Nous avons réalisé plusieurs expériences sur l'utilisation de notre modèle de détection d'analogie pour modéliser plusieurs langues à la fois (voir Subsection 8.1.2) et pour transférer le modèle entre les langues sans re-entraînement (voir Subsection 8.1.1 and Section 8.2), en tirant parti de la nature multilingue de Siganalogies.

Notre première tentative pour étudier le lien entre les performances de transfert et la famille a été réalisée dans [**transfert:2021:alsaidi**]. Cependant, les premières expériences sur Sig16 et JBATS ne nous ont pas permis de tirer des conclusions générales au-delà de la confirmation des limitations causées par les différences d'alphabets entre les langues. Nous avons également observé que le déséquilibre causé par l'augmentation des données sans échantillonnage se révèle lors du transfert vers d'autres langues, même lorsque les performances dans la langue d'apprentissage semblent satisfaisantes.

Among other results, with the cross-lingual transfer experiments performed on Sig19, we confirmed that under the right circumstances, i.e., when the alphabet gap is not limiting anymore, the morphological similarities between languages are reflected in the behavior of CNN+ANNc during transfer. By extension, the morphological transformations modeled by the analogy detection model through APs appear transferrable across languages. Nevertheless, the transfer performance is most likely influenced by the extent of the morphology of a language present in the data, as

Sig19 does not represent the full morphology of each language. This might explain why transfer in some clusters of languages does not perform as well as in the largest cluster we studied.

Entre autres résultats, les expériences de transfert interlinguistique réalisées sur Sig19 ont confirmé que dans les bonnes circonstances, c'est-à-dire lorsque les différences d'alphabet ne sont plus limitantes, les similitudes morphologiques entre les langues se reflètent dans le comportement de du modèle de détection d'analogie au cours du transfert. Par extension, les transformations morphologiques modélisées par le modèle de détection des analogies par le biais des PAs semblent transférables d'une langue à l'autre. Néanmoins, les performances de transfert sont très probablement influencées par l'étendue de la morphologie d'une langue présente dans les données, car Sig19 ne représente pas la morphologie complète de chaque langue. Cela pourrait expliquer pourquoi le transfert dans certains groupes de langues n'est pas aussi performant que dans le plus grand groupe que nous avons étudié.

D.3.5 Chapitre 9 : conclusion de la Partie II et dissemination du framework ANN

Le Chapitre 9 résume nos contributions de la Partie II. Dans ce chapitre, nous détaillons aussi les moyens en place pour disséminer les outils et ressources développés au long de nos travaux.

Le framework ANN atteint des performances très élevées en matière de détection et de résolution d'analogies, dans une variété de contextes. ANNc pour la détection d'analogies et ANNr pour la résolution d'analogies atteignent des performances état de l'art cohérentes sur toutes les langues utilisées dans nos expériences, et surpassent les modèles de références Alea [LYZ09] et Kolmo [Mur+20] d'une précision allant jusqu'à 80% dans certains cas. Ces performances sont, dans une certaine mesure, transférables d'une langue à l'autre.

Les performances supérieures résultent, entre autres, de *(i)* une représentation des mots apprise à partir des données en vue de la manipulation des analogies, *(ii)* la capacité du modèle à intégrer les dépendances entre les dimensions des embeddings, et *(iii)* la flexibilité d'aller au-delà des formules arithmétiques arbitraires des PAs telles que 3CosAdd, 3CosMul, ou la règle du parallélogramme. Cependant, il existe un compromis connu entre la performance des modèles de DL et leur interprétabilité. En particulier, il est généralement difficile de comprendre pourquoi un tel modèle obtient un résultat particulier. Cela s'applique également à notre framework, mais des travaux ultérieurs pourraient fournir des garanties théoriques ou des méthodes empiriques pour remédier à cette limitation.

Le framework ANN exploite les propriétés et les intuitions des PAs dans la conception des modèles, mais aussi pour augmenter les données de manière à améliorer la performance du modèle et réfuire la sensibilité aux conditions initiales. Le modèle apprend également à être invariant en ce qui concerne les permutations, conformément aux axiomes des PAs. Dans les contextes où les PAs semblent inadaptées en raison de certains des postulats, l'augmentation des données peut être adaptée, comme nous le montrons dans la Section 7.6 pour la Permutation des Moyens. Nous proposons une généralisation de ce processus dans Appendix D, qui sera étayée par des expériences dans des travaux ultérieurs.

Dans l'ensemble, notre framework montre qu'il est possible d'obtenir des performances élevées lors de la manipulation des PAs au-delà des modèles arithmétiques sur les embeddings ou des approches conçues manuellement. L'approche peut être appliquée à d'autres types de données en suivant nos directives, comme on peut le voir dans [Als+22b; JCM23; LPR19; LPR21; Zer+22] et Chapters 10 and 11.

D.4 Partie III : au delà des analogies morphologiques

D.4.1 Chapitre 10 : application du framework ANN au TSV

Le Chapitre 10 résume les contributions faites dans [Zer+22], et suit le même contenu. Des expériences complémentaires et des descriptions détaillées sont disponibles dans la thèse de doctorat de Zervakis [Zer23].

La tâche de vérification du sens d'un mot (Target Sense Verification, TSV) est un type de tâche de désambiguïsation du sens des mots qui consiste à déterminer si le sens d'un mot dans un contexte donné (sens voulu) correspond à l'un des sens possibles du mot (sens cible). Trois éléments sont fournis pour cette tâche : d'une part, le mot dans son contexte ; d'autre part, une définition et des hypernymes (c.a.d, mots ayant un sens plus général que le mot cible, mais couvrant le sens de ce dernier) pour un sens possible du mot.

Dans [Zer+22], nous abordons TSV en combinant BERT et ANNc et en reformulant TSV comme une tâche de détection d'analogie. Nos expériences démontrent l'impact significatif sur la performance finale de la position de la définition et des hypernymes dans l'entrée de BERT, ainsi que la façon dont l'accent est mis sur les hypernymes. En outre, nous obtenons des résultats compétitifs sur le benchmark d'évaluation WiC-TSV [Bre+21]. Enfin, nous expérimentons avec et sans l'augmentation analogique des données de Section 6.3, et nous observons que l'augmentation analogique des données donne des performances comparables et atténue la dépendance de BERT à l'égard de l'encodage des entrées.

D.4.2 Chapitre 11 : application du framework ANN à FrameNet

Dans le Chapitre 11, nous adaptons le framework ANN développée dans la Partie II dans le contexte de la sémantique des cadres, en nous concentrant sur le problème d'annotation des roles sémantiques (Frame Semantic Role Labeling, FSRL) sur FN1.7 (la dernière version de FN au moment de la rédaction de cet article). Nous reformulons FSRL comme un problème de résolution d'analogies dans Section 11.2. Nos expériences, présentées dans Section 11.4, montrent que, sous certaines conditions, l'utilisation de la résolution d'analogie permet d'obtenir des résultats plus performants que les approches état de l'art sur FSRL, sans utiliser de mécanismes de codage ou de décodage sophistiqués et coûteux en termes de calcul (contrairement aux précédents états de l'art).

D.4.3 Chapitre 12 : CoAT et compétence d'un cas/d'une base de cas

Le Chapitre 12 résume les contributions dans [Mar+23] et inclut d'autres expériences qui n'ont pas été publiées au moment de la rédaction.

La méthode CoAT, introduite par Badra [Bad20], est une méthode de raisonnement à partir de cas (CBR) basée sur la notion de transfert analogique. La méthode est basée sur Γ, un indicateur de la complexité d'un jeu de données du point de vue du transfert analogique, et permet de répondre à des questions telles que : “Une mesure de similarité donnée est-elle plus appropriée qu'une autre (pour une tâche) ?” ou “Dans quelle mesure une solution est-elle compatible avec un problème, compte tenu d'un case base et de mesures de similarité ?”

Ce que nous proposons dans [Mar+23], basé sur l'indicateur CoAT (Γ), diffère des travaux précédents sur la maintenance de base de cas sur trois principaux aspects :

1. au lieu de se concentrer sur les relations locales, comme c'est souvent le cas en CBR, Γ est un indicateur qui prend en compte l'ensemble de la la base de cas ;
2. pour déterminer l'utilité d'un cas à des fins de maintenance, au lieu d'estimer les problèmes futurs uniquement à partir de la base da cas, nous utilisons un ensemble de cas distincts de la base de cas comme référence ;
3. les résultats que nous obtenons remettent en question l'hypothèse courante selon laquelle la suppression de cas entraînera une perte de performance, et illustrent le fait que la compression peut en fait améliorer les performances.

De plus, dans [Mar+23], nous avons présenté une formulation à base d'énergie pour CoAT et des moyens de mesurer la compétence d'une base de cas pour des tâches d'apprentissage automatique, telles que la prédiction et la classification de cas. Cette approche de la compétence diffère des approches antérieures proposées dans la littérature car elle repose sur l'optimisation d'un indicateur de compatibilité globale entre deux mesures de similarité, l'une sur l'espace des situations et l'autre sur l'espace des issues. En plus des résultats présentés dans [Mar+23], nous fournissons des bornes pour la valeur de la fonction d'énergie et les mesures de compétence, bien que ces bornes puissent ne pas être strictes.

Nous montrons empiriquement, dans la Section 12.4, que notre notion de compétence est étroitement liée à la performance pour une tâche de classification binaire à partir de cas, dans le sens où la compétence d'un cas source est positivement corrélée à sa capacité à réduire l'énergie de l'issue correcte et à augmenter l'énergie de l'issue incorrecte. Nous analysons quantitativement et qualitativement le comportement d'un algorithme de compression basé sur les mesures de compétence proposées, sur différents ensembles de données avec des distributions substantiellement différentes et en prenant en compte différentes frontières de classification et critères de compression. En outre, nous analysons la robustesse de l'approche par rapport à différents ensembles de références et de cas initiaux. Une observation très encourageante des expériences dans Subsection 12.4.2 est que, lorsque l'on considère C_{hinge} comme la mesure de compétence et le critère de compression, la compétence commence à diminuer au même moment que la performance. Basé sur ces résultats, des expériences additionnelles et des garanties théoriques permettraient de concevoir un critère d'arrêt pour l'algorithme de compression de base de cas.

D.5 Conclusion

Dans cette thèse, nous présentons nos contributions sur le framework ANN sur la morphologie [Als+21a; Als+21b; Als+21c; Cha+22; Mar+22a; MC24; Mar+22b] dans la Partie II, tandis que la Partie III regroupe nos contributions pour la tâche de TSV [Zer+22], la sémantique des cadres, et pour le raisonnement à partir de cas [Bad+23; Mar+23]. Le Chapitre 13 résume ces contributions.

Le framework ANN sur la morphologie des mots. Dans la Partie II, nous avons proposé le framework ANN pour traiter la détection et la résolution d'analogies sur des PAs dont les éléments sont des mots et les relations sous-jacentes sont de nature morphologique.

Les modèles de manipulation d'analogie, appelés ANNc pour la détection d'analogie et ANNr pour la résolution d'équations analogiques, dépassent la performance des approches symboliques état de l'art (Nlg, Alea et Kolmo) et celle des approches de manipulation de vecteurs d'embedding (règle du parallélogramme, 3CosMul) dans la majorité des cas. En particulier, ANNc et ANNr sont en mesure de traiter des quadruplets qui ne correspondent pas exactement aux postulats des PAs, et les deux modèles d'embedding que nous proposons (basés respectivement sur l'architecture CNN et sur le principe d'auto-encodeur) sont à même de traiter les propriétés morphologiques dépendantes du contexte. Cela permet à nos modèles de traiter des PAs et des transformations morphologiques que les approches symboliques ont du mal à traiter.

Nous avons proposé Siganalogies comme jeu de données de PAs morphologiques dans plus de 80 langues, ce qui nous a permis d'effectuer des études comparatives intéressantes et de tester extensivement le framework ANN. Nous avons utilisé un protocole d'augmentation de données qui traduit les postulats des PAs afin d'entraîner des modèles d'apprentissage profond en utilisant des données du jeu de données Siganalogies. En conséquence, il est possible d'affirmer que la notion de PA modélisée par nos modèles est l'intersection des relations morphologiques présentes dans les données et des postulats des PAs. En particulier, nous avons été en mesure de montrer qu'altérer l'augmentation de données pour considérer différents ensembles de postulats résulte en des modèles correspondant à ces nouveaux ensembles de postulats. Cela nous a mené à étudier les combinaisons possibles de postulats pour pouvoir adapter le framework ANN à différents cas d'utilisation, détaillés dans le Chapitre d'Appendice D.

Le jeu de données Siganalogies et ses nombreuses langues nous ont permis d'explorer la façon dont ANNc couplé au modèle d'embedding utilisant un CNN (CNN+ANNc) se transfère entre différents domaines de données (c.a.d, différentes langues) qui partagent des mécanismes sous-jacents (c.a.d, les mécanismes des transformations morphologiques et les postulats des PAs). Parmi d'autres résultats, en entraînant sur plusieurs langues à la fois, nous avons pu augmenter la stabilité et la transférabilité du modèle entre les langues, sans augmenter le nombre d'exemples d'entraînement. Nous avons aussi corrélé la performance de CNN+ANNc avec la hiérarchie obtenue à partir des familles de langues.

En complément de ces travaux, nous avons effectué plusieurs études d'ablation et d'autres expériences pour améliorer le processus d'augmentation de données et la procédure d'entraînement de nos modèles.

Le framework ANN sur des phrases et de la sémantique. Suivant le succès de notre approche sur la morphologie, nous avons appliqué le processus d'augmentation de données et ANNc sur des embeddings produit par des gros modèles de langue (Large Language Models, PLMs), pour traiter deux applications sémantiques impliquant des phrases.

Premièrement, comme décrit dans le Chapitre 10, ANNc à été en mesure d'obtenir une performance état de l'art sur la tâche de TSV. Ces travaux nous ont permis de nous attaquer à plusieurs défis liés à la reformulation d'une tâche de classification (la tâche de TSV) sous forme d'une tâche de détection d'analogie, et liés à la manipulations de PAs hétérogènes où les éléments sont de nature différente.

Deuxièmement, dans le Chapitre 11, nous avons utilisé un modèle léger (une paire de Perceptrons manipulant des embeddings produits par mBert) pour transférer des annotations de sémantique des cadres d'une phrase à une autre en utilisant la résolution d'équation analogiques. De nouveau, nous avons dû reformuler le problème à traiter (la tâche de FSRL) sous forme d'équations analogiques. Notre modèle simple à été en mesure de dépasser la performance de l'état de l'art, un approche qui utilise un modèle significativement plus complexe, sans connaissance explicite des annotations manipulées. Cependant, cela n'a été possible qu'à condition d'utiliser la phrase source la plus adaptée pour le transfert analogique, un défi qui reste ouvert dans nos travaux.

Transfert analogique appliqué au raisonnement à partir de cas. Enfin, comme décrit dans le Chapitre 12, nous avons étendu l'approche CoAT à la compétence de (base de) cas, et à la compression de base de cas. Plus spécifiquement, nous avons défini des mesures de la compétence (qui peut être vue comme la pertinence ou le bénéfice pour une tâche donnée) à différent niveaux de granularité (compétence d'un cas, d'une base de cas, contribution d'un cas à la prédiction d'n cas de référence donné).

En utilisant ces mesures de compétence, nous avons défini un protocole simple pour la compression de base de cas, qui enlève itérativement le cas le moins compétent d'une base de cas, au regard d'un ensemble de cas de référence. En ôtant les cas qui ont un impact négatif sur la compétence globale d'une base de cas à partir des mesures que nous avons définies, nous avons obtenu une combinaison de la réduction de la taille de la base de cas avec une impressionnante amélioration de la performance prédictive de CoAT.

Nous avons aussi effectué des études d'ablation sur des données synthétiques, qui nous ont permis de démontrer la versatilité de l'approche (en particulier sa capacité à gérer des frontières de decision complexes) et de déterminer les spécificités du comportement du système dans des cas limites (par exemple, un déséquilibre dans la répartition des données).

References

[Afa+21] Afantenos, Stergos D., Kunze, Tarek, Lim, Suryani, Prade, Henri, and Richard, Gilles. “Analogies Between Sentences: Theoretical Aspects - Preliminary Experiments”. In: *ECSQARU*. Vol. 12897. LNCS. Springer, 2021, pp. 3–18. DOI: 10.1007/978-3-030-86772-0_1.

[Afa+22] Afantenos, Stergos D., Lim, Suryani, Prade, Henri, and Richard, Gilles. “Theoretical Study and Empirical Investigation of Sentence Analogies”. In: *IARML@IJCAI-ECAI*. Vol. 3174. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 15–28. URL: http://ceur-ws.org/Vol-3174/paper2.pdf.

[Aha92] Aha, David W. “Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms”. In: *Int. J. Man-Mach. Stud.* 36.2 (1992), pp. 267–287. ISSN: 0020-7373. DOI: https://doi.org/10.1016/0020-7373(92)90018-G.

[Ake+19] Aken, Betty van, Winter, Benjamin, Löser, Alexander, and Gers, Felix A. “How Does BERT Answer Questions? A Layer-Wise Analysis of Transformer Representations”. In: *CIKM*. Beijing, China: ACM, 2019, pp. 1823–1832. ISBN: 9781450369763. DOI: 10.1145/3357384.3358028.

[Ako18] Akoglu, Haldun. “User’s guide to correlation coefficients”. In: *Turk J Emerg Med* 18 (2018). DOI: 10.1016/j.tjem.2018.08.001.

[Ale+09] Alegria, Iñaki, Etxeberria, Izaskun, Hulden, Mans, and Maritxalar, Montserrat. “Porting Basque Morphological Grammars to Foma, an Open-Source Tool”. In: *FSMNLP*. Pretoria, South Africa: Springer-Verlag, 2009, pp. 105–113. ISBN: 364214683X.

[Als+22a] Alsaidi, Safa, Couceiro, Miguel, Marquer, Esteban, Quennelle, Sophie, Burgun, Anita, Garcelon, Nicolas, and Coulet, Adrien. “An analogy based framework for patient-stay identification in healthcare”. In: *ATA@ICCBR*. 2022. URL: https://inria.hal.science/hal-03763772.

[Als+22b] Alsaidi, Safa, Couceiro, Miguel, Quennelle, Sophie, Burgun, Anita, Garcelon, Nicolas, and Coulet, Adrien. “Exploring Analogical Inference in Healthcare”. In: *IARML@IJAI-ECAI*. Vol. 3174. CEUR Workshop Proceedings. 2022, pp. 40–50.

[Als+21a] Alsaidi, Safa, Decker, Amandine, Lay, Puthineath, Marquer, Esteban, Murena, Pierre-Alexandre, and Couceiro, Miguel. “A Neural Approach for Detecting Morphological Analogies”. In: *DSAA*. 2021, pp. 1–10.

[Als+21b] Alsaidi, Safa, Decker, Amandine, Lay, Puthineath, Marquer, Esteban, Murena, Pierre-Alexandre, and Couceiro, Miguel. “On the Transferability of Neural Models of Morphological Analogies”. In: *AIMLAI@ECML-PKDD*. Vol. 1524. 2021, pp. 76–89.

[Als+21c] Alsaidi, Safa, Decker, Amandine, Marquer, Esteban, Murena, Pierre-Alexandre, and Couceiro, Miguel. “Tackling Morphological Analogies Using Deep Learning - Extended Version”. In: *CoRR* (2021).

[Ant22] Antić, Christian. “Analogical Proportions”. In: *AMAI* 90.6 (2022), pp. 595–644. DOI: 10.1007/s10472-022-09798-y.

[Ant23] Antić, Christian. *Analogical Proportions in Undirected Graphs Via Path Justifications*. 2023. DOI: 10.2139/ssrn.4411400.

[Ara+21] Araki, Tomoyuki, Couceiro, Miguel, Drechsler, Rolf, Dubrova, Elena, Dueck, Gerhard, Gaudet, Vincent, Hirayama, Takashi, Ito, Akira, Kamide, Norihiro, Kamiura, Naotake, et al. “ISMVL2021 Reviewers”. In: *ISMVL* (2021). URL: https://ieeexplore.ieee.org/ielx7/9459630/9459631/09459670.pdf.

[AAB20] Ataman, Duygu, Aziz, Wilker, and Birch, Alexandra. “A Latent Morphology Model for Open-Vocabulary Neural Machine Translation”. In: *ICLR*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=BJxSI1SKDH.

[Bad20] Badra, Fadi. “A Dataset Complexity Measure for Analogical Transfer”. In: *IJCAI*. 2020, pp. 1601–1607. DOI: 10.24963/ijcai.2020/222.

[BL22] Badra, Fadi and Lesot, Marie-Jeanne. “CoAT-APC: When Analogical Proportion-based Classification Meets Case-based Prediction”. In: *ATA@ICCBR*. 2022.

[BL23] Badra, Fadi and Lesot, Marie-Jeanne. “Case-based prediction - A survey”. In: *Int. J. Approx. Reason.* 158 (2023), p. 108920. DOI: 10.1016/j.ijar.2023.108920.

[Bad+22] Badra, Fadi, Lesot, Marie-Jeanne, Barakat, Aman, and Marsala, Christophe. “Theoretical and Experimental Study of a Complexity Measure for Analogical Transfer”. In: *ICCBR*. 2022.

[Bad+23] Badra, Fadi, Lesot, Marie-Jeanne, Marquer, Esteban, and Couceiro, Miguel. “Some perspectives on similarity learning for case-based reasoning and analogical transfer”. In: *IARML@IJCAI*. 2023.

[Bae+22] Baevski, Alexei, Hsu, Wei-Ning, Xu, Qiantong, Babu, Arun, Gu, Jiatao, and Auli, Michael. “Data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language”. In: *ai.meta.com* (2022). URL: https://ai.meta.com/research/data2vec-a-general-framework-for-self-supervised-learning-in-speech-vision-and-language.

[BCB15] Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *ICLR*. 2015. URL: http://arxiv.org/abs/1409.0473.

[Bak17] Baker, Collin F. “Framenet: Frame semantic annotation in practice”. In: *HLA* (2017), pp. 771–811. DOI: 10.1007/978-94-024-0881-2_28.

[BFL98] Baker, Collin F., Fillmore, Charles J., and Lowe, John B. “The Berkeley FrameNet Project”. In: *COLING*. Vol. 1. ACL, 1998, pp. 86–90. DOI: 10.3115/980845.980860. URL: https://aclanthology.org/P98-1013.

[BMP19] Barbot, Nelly, Miclet, Laurent, and Prade, Henri. “Analogy between concepts”. In: *AI* 275 (2019), pp. 487–539.

[BPL22] Bardes, Adrien, Ponce, Jean, and LeCun, Yann. “VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning”. In: *ICLR*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=xm6YD62D1Ub.

[Bat+22] Batsuren, Khuyagbaatar, Bella, Gábor, Arora, Aryaman, Martinovic, Viktor, Gorman, Kyle, Žabokrtský, Zdeněk, Ganbold, Amarsanaa, Dohnalová, Šárka, Ševčíková, Magda, Pelegrinová, Kateřina, Giunchiglia, Fausto, Cotterell, Ryan, and Vylomova, Ekaterina. “The SIGMORPHON 2022 Shared Task on Morpheme Segmentation”. In: *SIGMORPHON*. ACL, 2022, pp. 103–116. DOI: 10.18653/v1/2022.sigmorphon-1.11.

[BMD07] Bayoudh, Sabri, Miclet, Laurent, and Delhay, Arnaud. “Learning by Analogy: A Classification Rule for Binary and Nominal Data”. In: *IJCAI*. 2007, pp. 678–683. URL: http://ijcai.org/Proceedings/07/Papers/108.pdf.

[Bay+07] Bayoudh, Sabri, Mouchère, Harold, Miclet, Laurent, and Anquetil, Éric. “Learning a Classifier with Very Few Examples: Analogy Based and Knowledge Based Generation of New Examples for Character Recognition”. In: *ECML*. Vol. 4701. LNCS. Springer, 2007, pp. 527–534. DOI: 10.1007/978-3-540-74958-5_49.

[Ben12] Bengio, Yoshua. “Practical Recommendations for Gradient-Based Training of Deep Architectures”. In: *Neural Networks: Tricks of the Trade - Second Edition*. Vol. 7700. LNCS. Springer, 2012, pp. 437–478. DOI: 10.1007/978-3-642-35289-8_26.

[Bit+23] Bitton, Yonatan, Yosef, Ron, Strugo, Eliyahu, Shahaf, Dafna, Schwartz, Roy, and Stanovsky, Gabriel. “VASR: Visual Analogies of Situation Recognition”. In: *AAAI-IAAI-EAAI*. AAAI Press, 2023, pp. 241–249. DOI: 10.1609/AAAI.V37I1.25096.

[Bod09] Bod, Rens. “From exemplar to grammar: A probabilistic analogy-based model of language learning”. In: *CogSci* 33.5 (2009), pp. 752–793.

[Boj+17] Bojanowski, Piotr, Grave, Edouard, Joulin, Armand, and Mikolo, Tomás. “Enriching Word Vectors with Subword Information”. In: *ACL* 5 (2017), pp. 135–146.

[BJS18] Bouraoui, Zied, Jameel, Shoaib, and Schockaert, Steven. “Relation Induction in Word Embeddings Revisited”. In: *COLING*. ACL, 2018, pp. 1627–1637. URL: https://aclanthology.org/C18-1138.

[Bra81] Brakel, Arthur. “Boundaries in a morphological grammar of Portuguese”. In: *WORD* 32.3 (1981), pp. 193–212.

[Bre+21] Breit, Anna, Revenko, Artem, Rezaee, Kiamehr, Pilehvar, Mohammad Taher, and Camacho-Collados, José. “WiC-TSV: An Evaluation Benchmark for Target Sense Verification of Words in Context”. In: *EACL*. 2021, pp. 1635–1645. DOI: 10.18653/v1/2021.eacl-main.140.

[Cam13] Camilleri, John J. “A Computational Grammar and Lexicon for Maltese”. Master’s tehsis. Chalmers University of Technology., 2013.

[CH23] Canby, Marc and Hockenmaier, Julia. “A Framework for Bidirectional Decoding: Case Study in Morphological Inflection”. In: *EMNLP*. ACL, 2023, pp. 4485–4507. DOI: 10.18653/v1/2023.findings-emnlp.297.

[CR16] Cao, Kris and Rei, Marek. “A Joint Model for Word Embedding and Word Morphology”. In: *Rep4NLP@ACL*. 2016. URL: https://api.semanticscholar.org/CorpusID:7511759.

[Cha+22] Chan, Kevin, Kaszefski-Yaschuk, Shane Peter, Saran, Camille, Marquer, Esteban, and Couceiro, Miguel. “Solving Morphological Analogies Through Generation”. In: *IARML@IJCAI-ECAI*. Vol. 3174. 2022, pp. 29–39.

[CV03] Chater, Nick and Vitányi, Paul. “Simplicity: a unifying principle in cognitive science?” In: *Trends Cog. Sci.* 7.1 (2003), pp. 19–22. ISSN: 1364-6613. DOI: 10.1016/S1364-6613(02)00005-0.

[CPG17] Chen, Dawn, Peterson, Joshua C., and Griffiths, Tom. “Evaluating vector-space models of analogy”. In: *CogSci*. 2017, pp. 1746–1751.

[Cho17] Chollet, François. *Character-level recurrent sequence-to-sequence model*. 2017. URL: https://keras.io/examples/nlp/lstm_seq2seq.

[Chu+19] Chuang, Yu-Ying, Lõo, Kaidi, Blevins, J., and Baayen, R. Harald. “Estonian case inflection made simple. A case study in Word and Paradigm morphology with Linear Discriminative Learning.” In: *Complex Words: Advances in Morphology*. 2019, pp. 119–41. URL: https://api.semanticscholar.org/CorpusID:242395921.

[Cor96] Cornuéjols, Antoine. “Analogy as Minimization of Description Length”. In: (1996), pp. 321–335. URL: https://hal.science/hal-02480316.

[CA98] Cornuéjols, Antoine and Ales-Bianchetti, Jacques. “Analogy and Induction: which (missing) link?” In: *AAR-ITDCCNS*. 1998.

[CV95] Cortes, Corinna and Vapnik, Vladimir. “Support-vector networks”. In: *ML* 20.3 (1995), pp. 273–297.

[Cot+17] Cotterell, Ryan, Kirov, Christo, Sylak-Glassman, John, Walther, Géraldine, Vylomova, Ekaterina, Xia, Patrick, Faruqui, Manaal, Kübler, Sandra, Yarowsky, David, Eisner, Jason, and Hulden, Mans. “CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection in 52 Languages”. In: *CoNLL SIGMORPHON*. ACL, 2017, pp. 1–30. DOI: 10.18653/v1/K17-2001.

[Cot+16] Cotterell, Ryan, Kirov, Christo, Sylak-Glassman, John, Yarowsky, David, Eisner, Jason, and Hulden, Mans. “The SIGMORPHON 2016 Shared Task–Morphological Reinflection”. In: *SIGMORPHON*. ACL, 2016, pp. 10–22.

[CSE16] Cotterell, Ryan, Schütze, Hinrich, and Eisner, Jason. “Morphological Smoothing and Extrapolation of Word Embeddings”. In: *ACL*. ACL, 2016, pp. 1651–1660. DOI: 10.18653/v1/P16-1156.

[Cou+17a] Couceiro, Miguel, Hug, Nicolas, Prade, Henri, and Richard, Gilles. "Analogy-preserving Functions: A Way to Extend Boolean Samples". In: *IJCAI*. 2017, pp. 1–12.

[Cou+17b] Couceiro, Miguel, Hug, Nicolas, Prade, Henri, and Richard, Gilles. "Analogy-preserving functions: A way to extend Boolean samples". In: *IJCAI*. 2017, pp. 1575–1581.

[Cou+18] Couceiro, Miguel, Hug, Nicolas, Prade, Henri, and Richard, Gilles. "Behavior of Analogical Inference w.r.t. Boolean Functions". In: *IJCAI*. 2018, pp. 2057–2063.

[CL24] Couceiro, Miguel and Lehtonen, Erkko. "Galois theory for analogical classifiers". In: *AMAI* 92.1 (2024), pp. 29–47. DOI: 10.1007/S10472-023-09833-6.

[DBF21] Dannélls, Dana, Borin, Lars, and Friberg Heppin, Karin. *The Swedish FrameNet++ Harmonization, integration, method development and practical language technology applications*. John Benjamins Publishing Company, 2021. ISBN: 9789027209900.

[Dek+12] Dekel, Ofer, Gilad-Bachrach, Ran, Shamir, Ohad, and Xiao, Lin. "Optimal Distributed Online Prediction Using Mini-Batches". In: *J. Mach. Learn. Res.* 13 (2012), pp. 165–202. DOI: 10.5555/2503308.2188391.

[DL23] Deng, Xulin and Lepage, Yves. "Resolution of Analogies Between Strings in the Case of Multiple Solutions". In: *ATA@ICCBR*. Vol. 3438. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 3–14. URL: https://ceur-ws.org/Vol-3438/paper%5C_01.pdf.

[Den+88] Denker, John S., Gardner, William R., Graf, Hans Peter, Henderson, Donnie, Howard, Richard E., Hubbard, Wayne E., Jackel, Lawrence D., Baird, Henry S., and Guyon, Isabelle. "Neural Network Recognizer for Hand-Written Zip Code Digits". In: *NIPS*. Morgan Kaufmann, 1988, pp. 323–331. URL: http://papers.nips.cc/paper/107-neural-network-recognizer-for-hand-written-zip-code-digits.

[Dev+19] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL-HLT*. Vol. 1. ACL, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.

[DZF19] Diallo, Aïssatou, Zopf, Markus, and Fürnkranz, Johannes. "Learning Analogy-Preserving Sentence Embeddings for Answer Selection". In: *CoNLL*. ACL, 2019, pp. 910–919. DOI: 10.18653/V1/K19-1085.

[DWW22] Didi, Yosra, Walha, Ahlam, and Wali, Ali. "COVID-19 Tweets Classification Based on a Hybrid Word Embedding Method". In: *BDCC* 6 (2022), p. 58. DOI: 10.3390/bdcc6020058.

[Dje+16] Djemaa, Marianne, Candito, Marie, Muller, Philippe, and Vieu, Laure. "Corpus Annotation within the French FrameNet: a Domain-by-domain Methodology". In: *LREC*. ELRA, 2016, pp. 3794–3801. URL: https://aclanthology.org/L16-1601.

[Doz16] Dozat, Timothy. "Incorporating Nesterov Momentum into Adam". In: (2016).

[DGM16] Drozd, Aleksandr, Gladkova, Anna, and Matsuoka, Satoshi. "Word Embeddings, Analogies, and Machine Learning: Beyond king - man + woman = queen". In: *COLING*. ACL, 2016, pp. 3519–3530. URL: https://aclanthology.org/C16-1332.

[Dum+88] Dumais, Susan T, Furnas, George W, Landauer, Thomas K, Deerwester, Scott, and Harshman, Richard. "Using latent semantic analysis to improve access to textual information". In: *SIGCHI*. 1988, pp. 281–285.

[DV16] Dumoulin, Vincent and Visin, Francesco. "A guide to convolution arithmetic for deep learning". In: *CoRR* abs/1603.07285 (2016). arXiv: 1603.07285. URL: http://arxiv.org/abs/1603.07285.

[EL10] Eddington, David and Lachler, Jordan. "A Computational Analysis of Navajo Verb Stems". In: *EEMCFR*. 2010, pp. 143–161.

[Elm90] Elman, Jeffrey L. "Finding structure in time". In: *Cog. Sci.* 14.2 (1990), pp. 179–211. ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E.

[FL16] Fam, Rashel and Lepage, Yves. "Morphological Predictability of Unseen Words Using Computational Analogy". In: *ICCBR*. Vol. 1815. CEUR Workshop Proceedings. CEUR-WS.org, 2016, pp. 51–60. URL: https://ceur-ws.org/Vol-1815/paper5.pdf.

[FL18] Fam, Rashel and Lepage, Yves. “Tools for The Production of Analogical Grids and a Resource of N-gram Analogical Grids in 11 Languages”. In: *LREC*. ELRA, 2018, pp. 1060–1066.

[Gal16] Gal, Yarin. “Uncertainty in Deep Learning”. PhD thesis. University of Cambridge, 2016.

[Gen83] Gentner, Dedre. “Structure Mapping: A Theoretical Framework for Analogy”. In: *Cog. Sci.* 7 (1983), pp. 155–170. DOI: 10.1207/s15516709cog0702_3.

[GHK01] Gentner, Dedre, Holyoak, Keith J., and Kokinov, Boicho N. *The analogical mind: Perspectives from cognitive science.* MIT Press, 2001. DOI: 10.7551/mitpress/1251.001.0001.

[Gir23] Girrbach, Leander. “Tü-CL at SIGMORPHON 2023: Straight-Through Gradient Estimation for Hard Attention”. In: *SIGMORPHON*. ACL, 2023, pp. 171–185. DOI: 10.18653/v1/2023.sigmorphon-1.19.

[GDM16] Gladkova, Anna, Drozd, Aleksandr, and Matsuoka, Satoshi. “Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn’t”. In: *NAACL*. ACL, 2016, pp. 8–15. DOI: 10.18653/V1/N16-2002.

[GBB11] Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. “Deep Sparse Rectifier Neural Networks”. In: *AISTATS*. Vol. 15. JMLR Proceedings. JMLR.org, 2011, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf.

[Gol+23] Goldman, Omer, Batsuren, Khuyagbaatar, Khalifa, Salam, Arora, Aryaman, Nicolai, Garrett, Tsarfaty, Reut, and Vylomova, Ekaterina. “SIGMORPHON–UniMorph 2023 Shared Task 0: Typologically Diverse Morphological Inflection”. In: *SIGMORPHON*. ACL, 2023, pp. 117–125. DOI: 10.18653/v1/2023.sigmorphon-1.13.

[GS05] Graves, Alex and Schmidhuber, Jürgen. “Framewise phoneme classification with bidirectional LSTM and other neural network architectures”. In: *NN* 18.5-6 (2005), pp. 602–610.

[Gus+08] Gust, Helmar, Krumnack, Ulf, Kühnberger, Kai-Uwe, and Schwering, Angela. “Analogical Reasoning: A Core of Cognition”. In: *Künstliche Intelligenz* 22.1 (2008), pp. 8–12. URL: http://www.kuenstliche-intelligenz.de/fileadmin/template/main/archiv/pdf/ki2008-01%5C_page8%5C_web%5C_teaser.pdf.

[Hat08] Hathout, Nabil. “Acquistion of the Morphological Structure of the Lexicon Based on Lexical Similarity and Formal Analogy”. In: *TextGraphs@COLING*. 2008, pp. 1–8. URL: https://aclanthology.org/W08-2001.

[HLZ15] He, Luheng, Lewis, Mike, and Zettlemoyer, Luke. “Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language”. In: *EMNLP*. ACL, 2015, pp. 643–653. DOI: 10.18653/v1/D15-1076. URL: https://aclanthology.org/D15-1076.

[Hin+06] Hinton, Geoffrey, Osindero, Simon, Welling, Max, and Teh, Yee-Whye. “Unsupervised Discovery of Nonlinear Structure Using Contrastive Backpropagation”. In: *Cog. Sci.* 30.4 (2006), pp. 725–731. ISSN: 03640213.

[Ho95] Ho, Tin Kam. “Random decision forests”. In: *ICDAR*. Vol. 1. 1995, pp. 278–282.

[HB00] Ho, Tin Kam and Basu, M. “Measuring the complexity of classification problems”. In: *ICPR*. Vol. 2. 2000, 43–47 vol.2. DOI: 10.1109/ICPR.2000.906015.

[HS96] Hochreiter, Sepp and Schmidhuber, Jürgen. “LSTM can Solve Hard Long Time Lag Problems”. In: *NIPS*. 1996, pp. 473–479.

[HM95] Hofstadter, Douglas and Mitchell, Melanie. “The copycat project: A model of mental fluidity and analogy-making”. In: *FCCAs*. 1995. Chap. 5, pp. 205–267.

[Hue+21] Huembeli, Patrick, Arrazola, Juan Miguel, Killoran, Nathan, Mohseni, Masoud, and Wittek, Peter. *The Physics of Energy-Based Models - Towards Data Science.* 2021. URL: https://towardsdatascience.com/the-physics-of-energy-based-models-1121122d0d9.

[Hug+19] Hug, Nicolas, Prade, Henry, Richard, Gilles, and Serrurier, Mathieu. "Analogical proportion-based methods for recommendation - First investigations". In: *Fuzzy Sets Syst.* 366 (2019), pp. 110–132. ISSN: 0165-0114. DOI: https://doi.org/10.1016/j.fss.2018.11.007.

[HGS13] Hwang, Sung Ju, Grauman, Kristen, and Sha, Fei. "Analogy-preserving Semantic Embedding for Visual Object Categorization". In: *ICML*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 639–647. URL: http://proceedings.mlr.press/v28/juhwang13.html.

[JSS23] Jacob, Shahar, Shani, Chen, and Shahaf, Dafna. "FAME: Flexible, Scalable Analogy Mappings Engine". In: *EMNLP*. ACL, 2023, pp. 16426–16442. URL: https://aclanthology.org/2023.emnlp-main.1023.

[JCM23] Jarnac, Lucas, Couceiro, Miguel, and Monnin, Pierre. "Relevant Entity Selection: Knowledge Graph Bootstrapping via Zero-Shot Analogical Pruning". In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023, Birmingham, United Kingdom, October 21-25, 2023*. ACM, 2023, pp. 934–944. DOI: 10.1145/3583780.3615030.

[KAÜ19] Karabulut, Bergen, Arslan, Güvenç, and Ünver, Halil Murat. "A Weighted Similarity Measure for k-Nearest Neighbors Algorithm". In: *CBUJOS* 15.4 (2019), pp. 393–400. DOI: 10.18466/cbayarfbe.618964.

[Kar+18] Karpinska, Marzena, Li, Bofang, Rogers, Anna, and Drozd, Aleksandr. "Subcharacter Information in Japanese embeddings: when is it worth it?" In: *RELNLP*. ACL, 2018, pp. 28–37.

[KS20] Keane, Mark T. and Smyth, Barry. "Good Counterfactuals and Where to Find Them: A Case-Based Technique for Generating Counterfactuals for Explainable AI (XAI)". In: *Case-Based Reasoning Research and Development - 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings*. Vol. 12311. LNCS. Springer, 2020, pp. 163–178. DOI: 10.1007/978-3-030-58342-2_11.

[Kie10] Kiefer, Ferenc. "Hungarian". In: *Revue belge de philologie et d'histoire* 88 (2010).

[KB15] Kingma, Diederik P. and Ba, Jimmy. "Adam: A Method for Stochastic Optimization". In: *ICLR*. 2015.

[Kir+18] Kirov, Christo, Cotterell, Ryan, Sylak-Glassman, John, Walther, Géraldine, Vylomova, Ekaterina, Xia, Patrick, Faruqui, Manaal, Mielke, Sabrina J., McCarthy, Arya, Kübler, Sandra, Yarowsky, David, Eisner, Jason, and Hulden, Mans. "UniMorph 2.0: Universal Morphology". In: *LREC*. ELRA, 2018. URL: https://aclanthology.org/L18-1293.

[Kir+16] Kirov, Christo, Sylak-Glassman, John, Que, Roger, and Yarowsky, David. "Very-large Scale Parsing and Normalization of Wiktionary Morphological Paradigms". In: *LREC*. ELRA, 2016, pp. 3121–3126. URL: https://aclanthology.org/L16-1498.

[Kra91] Kramer, Mark A. "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE* 37.2 (1991), pp. 233–243.

[Kus+17] Kusner, Matt J., Loftus, Joshua, Russell, Chris, and Silva, Ricardo. "Counterfactual Fairness". In: *NIPS*. Vol. 30. Curran Associates, Inc., 2017.

[KHW23] Kwak, Alice, Hammond, Michael, and Wing, Cheyenne. "Morphological reinflection with weighted finite-state transducers". In: *SIGMORPHON*. ACL, 2023, pp. 132–137. DOI: 10.18653/v1/2023.sigmorphon-1.15.

[LYZ09] Langlais, Philippe, Yvon, François, and Zweigenbaum, Pierre. "Improvements in Analogical Learning: Application to Translating Multi-Terms of the Medical Domain". In: *EACL*. ACL, 2009, pp. 487–495.

[LTC17] Law, Marc T., Thome, Nicolas, and Cord, Matthieu. "Learning a Distance Metric from Relative Comparisons between Quadruplets of Images". In: *Int. J. Comput. Vis.* 121.1 (2017), pp. 65–94. DOI: 10.1007/S11263-016-0923-4.

[LeC+89] LeCun, Yann, Boser, Bernhard E., Denker, John S., Henderson, Donnie, Howard, Richard E., Hubbard, Wayne E., and Jackel, Lawrence D. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *NeurComp* 1.4 (1989), pp. 541–551.

[LeC+06] LeCun, Yann, Chopra, Sumit, Hadsell, Raia, Ranzato, Marc'Aurelio, and Huang, Fu Jie. "A Tutorial on Energy-Based Learning". In: *Predicting Structured Data*. MIT Press, 2006, p. 59.

[LÖ23] Leemhuis, Mena and Özçep, Özgür Lütfü. "Analogical Proportions and Betweenness". In: *Proceedings of the 9th Workshop on Formal and Cognitive Reasoning co-located with the 46th German Conference on Artificial Intelligence (KI 2023), Berlin, Germany, September 26, 2023*. Vol. 3500. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 8–19. URL: https://ceur-ws.org/Vol-3500/paper0.pdf.

[Lee79] Leer, Jeff. *Proto-Athabaskan Verb Stem Variation. Part One: Phonology.* 1979.

[Lep98] Lepage, Yves. "Solving analogies on words: an algorithm". In: *COLING*. ACL, 1998, pp. 728–734. DOI: 10.3115/980845.980967.

[Lep01] Lepage, Yves. "Analogy and Formal Languages". In: vol. 53. 2001, pp. 180–191. DOI: 10.1016/S1571-0661(05)82582-4.

[Lep03] Lepage, Yves. "De l'analogie rendant compte de la commutation en linguistique". FR. Habilitation à diriger des recherches. Université Joseph-Fourier - Grenoble I, 2003. URL: https://tel.archives-ouvertes.fr/tel-00004372.

[Lep14] Lepage, Yves. "Analogies Between Binary Images: Application to Chinese Characters". In: *Computational Approaches to Analogical Reasoning: Current Trends*. Vol. 548. Studies in Computational Intelligence. Springer, 2014, pp. 25–57. DOI: 10.1007/978-3-642-54516-0_2.

[Lep17] Lepage, Yves. "Character-Position Arithmetic for Analogy Questions between Word Forms". In: *CAW@ICCBR*. Vol. 2028. 2017, pp. 23–32.

[Lep19] Lepage, Yves. "Analogies Between Short Sentences: A Semantico-Formal Approach". In: *LTC*. Vol. 13212. LNCS. Springer, 2019, pp. 163–179. DOI: 10.1007/978-3-031-05328-3_11.

[LA96] Lepage, Yves and Ando, Shinichi. "Saussurian analogy: a theoretical account and its application". In: *COLING*. 1996, pp. 717–722.

[LD05] Lepage, Yves and Denoual, Etienne. "Purest ever example-based machine translation: Detailed presentation and assessment". In: *Mach. Transl.* 19.3-4 (2005), pp. 251–282. DOI: 10.1007/S10590-006-9010-X.

[LG14] Levy, Omer and Goldberg, Yoav. "Linguistic Regularities in Sparse and Explicit Word Representations". In: *CoNLL*. ACL, 2014, pp. 171–180. DOI: 10.3115/v1/W14-1618.

[Li+14] Li, Mu, Zhang, Tong, Chen, Yuqiang, and Smola, Alexander J. "Efficient mini-batch training for stochastic optimization". In: *KDD*. ACM, 2014, pp. 661–670. DOI: 10.1145/2623330.2623612.

[LNP21] Lieber, Jean, Nauer, Emmanuel, and Prade, Henri. "When Revision-Based Case Adaptation Meets Analogical Extrapolation". In: *ICCBR*. Vol. 12877. LNCS. 2021, pp. 156–170.

[LPR19] Lim, Suryani, Prade, Henri, and Richard, Gilles. "Solving Word Analogies: A Machine Learning Perspective". In: *ECSQARU*. Vol. 11726. Springer, 2019, pp. 238–250. DOI: 10.1007/978-3-030-29765-7_20.

[LPR21] Lim, Suryani, Prade, Henri, and Richard, Gilles. "Classifying and completing word analogies by machine learning". In: *IJAR* 132 (2021), pp. 1–25.

[LSZ21] Lin, ZhiChao, Sun, Yueheng, and Zhang, Meishan. "A Graph-Based Neural Model for End-to-End Frame Semantic Parsing". In: *EMNLP*. ACL, 2021, pp. 3864–3874. DOI: 10.18653/v1/2021.emnlp-main.314.

[Liu+21] Liu, Qianchu, Liu, Fangyu, Collier, Nigel, Korhonen, Anna, and Vulic, Ivan. "MirrorWiC: On Eliciting Word-in-Context Representations from Pretrained Language Models". In: *CoNLL*. 2021, pp. 562–574. DOI: 10.18653/v1/2021.conll-1.44.

[LPM15] Luong, Thang, Pham, Hieu, and Manning, Christopher D. "Effective Approaches to Attention-based Neural Machine Translation". In: *EMNLP*. ACL, 2015, pp. 1412–1421. DOI: 10.18653/V1/D15-1166.

[LSM13] Luong, Thang, Socher, Richard, and Manning, Christopher D. "Better Word Representations with Recursive Neural Networks for Morphology". In: *Conference on Computational Natural Language Learning*. 2013. URL: https://api.semanticscholar.org/CorpusID:14276764.

[24] *Machine Learning Models*. 2024. URL: https://www.javatpoint.com/machine-learning-models.

[Man19] Mansfield, John. *Murrinhpatha morphology and phonology*. De Gruyter Mouton Bosten ; Berlin, 2019. ISBN: 9781501511394.

[ML23] Mao, Weihao and Lepage, Yves. "Embedding-To-Embedding Method Based on Autoencoder for Solving Sentence Analogies". In: *ATA@ICCBR*. Vol. 3438. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 15–26. URL: https://ceur-ws.org/Vol-3438/paper%5C_02.pdf.

[Mar+22a] Marquer, Esteban, Alsaidi, Safa, Decker, Amandine, Murena, Pierre-Alexandre, and Couceiro, Miguel. "A Deep Learning Approach to Solving Morphological Analogies". In: *ICCBR*. Vol. 13405. LNCS. Springer, 2022, pp. 159–174. DOI: 10.1007/978-3-031-14923-8_11.

[Mar+23] Marquer, Esteban, Badra, Fadi, Lesot, Marie-Jeanne, Couceiro, Miguel, and Leake, David. "Less is Better: An Energy-Based Approach to Case Base Competence". In: *ICCBR-WS*. Vol. 3438. CEUR-WS.org, 2023, pp. 27–42. URL: https://ceur-ws.org/Vol-3438/paper%5C_03.pdf.

[MC24] Marquer, Esteban and Couceiro, Miguel. "Solving morphological analogies: from retrieval to generation". In: *AMAI:MFARA* (2024). arXiv: 2303.18062. URL: https://arxiv.org/abs/2303.18062.

[Mar+22b] Marquer, Esteban, Couceiro, Miguel, Alsaidi, Safa, and Decker, Amandine. *Siganalogies - morphological analogies from Sigmorphon 2016 and 2019*. Version V1. 2022.

[MMC22] Marquer, Esteban, Murena, Pierre-Alexandre, and Couceiro, Miguel. "Transferring Learned Models of Morphological Analogy". In: *ATA@ICCBR*. 2022.

[Mat17] Mattiello, Elisa. *Analogy in Word-formation*. 2017.

[McC+19] McCarthy, Arya D., Vylomova, Ekaterina, Wu, Shijie, Malaviya, Chaitanya, Wolf-Sonkin, Lawrence, Nicolai, Garrett, Kirov, Christo, Silfverberg, Miikka, Mielke, Sabrina J., Heinz, Jeffrey, Cotterell, Ryan, and Hulden, Mans. "The SIGMORPHON 2019 Shared Task: Morphological Analysis in Context and Cross-Lingual Transfer for Inflection". In: *CRPPM@ACL*. 2019, pp. 229–244.

[MP43] McCulloch, Warren S. and Pitts, Walter. "A logical calculus of the ideas immanent in nervous activity". In: *Bull. Math. Biophys.* 5.4 (1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259.

[MBD08] Miclet, Laurent, Bayoudh, Sabri, and Delhay, Arnaud. "Analogical Dissimilarity: Definition, Algorithms and Two Experiments in Machine Learning". In: *JAIR* 32 (2008), pp. 793–824. DOI: 10.1613/jair.2519. arXiv: 1401.3427.

[MP09a] Miclet, Laurent and Prade, Henri. "Handling Analogical Proportions in Classical Logic and Fuzzy Logics Settings". In: *ECSQARU*. Vol. 5590. LNCS. Springer, 2009, pp. 638–650. DOI: 10.1007/978-3-642-02906-6_55.

[MP09b] Miclet, Laurent and Prade, Henri. "Handling analogical proportions in classical logic and fuzzy logics settings". In: *ECSQARU*. Vol. 5590. LNCS. Springer, 2009, pp. 638–650. DOI: 10.1007/978-3-642-02906-6_55.

[MYZ13] Mikolov, Tomas, Yih, Wen-tau, and Zweig, Geoffrey. "Linguistic Regularities in Continuous Space Word Representations". In: *NAACL*. ACL, 2013, pp. 746–751. URL: https://aclanthology.org/N13-1090.

[Mik+13] Mikolov, Tomás, Kai, Chenn, Corrado, Gregory S., and Dean, Jeffrey. "Efficient Estimation of Word Representations in Vector Space". In: *ICLR*. 2013. DOI: 10.48550/ARXIV.1301.3781.

[Mit21] Mitchell, Melanie. "Abstraction and Analogy-Making in Artificial Intelligence". In: *ANYAS* 1505 (1 2021), pp. 79–101.

[MC22] Monnin, Pierre and Couceiro, Miguel. “Interactions Between Knowledge Graph-Related Tasks and Analogical Reasoning: A Discussion”. In: *ATA@ICCBR*. Vol. 3389. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 57–67. URL: https://ceur-ws.org/Vol-3389/ICCBR%5C_2022%5C_Workshop%5C_paper%5C_75.pdf.

[MPD21] Moreno, Jose G, Pontes, Elvys Linhares, and Dias, Gaël. “CTLR@WiC-TSV: Target Sense Verification using Marked Inputs and Pre-trained Models”. In: *SemDeep-6*. 2021, pp. 1–6.

[Mur22] Murena, Pierre-Alexandre. “Measuring the Feasibility of Analogical Transfer using Complexity”. In: *Proceedings of the Workshop on the Interactions between Analogical Reasoning and Machine Learning (International Joint Conference on Artificial Intelligence - European Conference on Artificial Intelligence (IJAI-ECAI 2022)), Vienna, Austria, July 23, 2022*. Vol. 3174. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 62–74. URL: https://ceur-ws.org/Vol-3174/paper6.pdf.

[MDC17] Murena, Pierre-Alexandre, Dessalles, Jean-Louis, and Cornuéjols, Antoine. “A complexity based approach for solving Hofstadter’s analogies”. In: *CAW@ICCBR*. 2017.

[Mur+20] Murena, Pierre-Alexandre, Al-Ghossein, Marie, Dessalles, Jean-Louis, and Cornuéjols, Antoine. “Solving Analogies on Words based on Minimal Complexity Transformation”. In: *IJCAI*. 2020, pp. 1848–1854.

[NF02] Neuvel, Sylvain and Fulop, Sean A. “Unsupervised Learning of Morphology Without Morphemes”. In: *SIGMORPHON*. ACL, 2002, pp. 31–40. DOI: 10.3115/1118647.1118651.

[Nie15] Nielsen, Michael A. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com.

[NR22] Nzeyimana, Antoine and Rubungo, Andre Niyongabo. “KinyaBERT: a Morphology-aware Kinyarwanda Language Model”. In: *ACL*. ACL, 2022, pp. 5347–5363. DOI: 10.18653/V1/2022.ACL-LONG.367.

[PSM14] Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. “GloVe: Global Vectors for Word Representation”. In: *EMNLP*. 2014, pp. 1532–1543.

[Per19] Perner, Petra. “Case-Based Reasoning – Methods, Techniques, and Applications”. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer International Publishing, 2019, pp. 16–30. ISBN: 978-3-030-33904-3.

[Pey+19] Peyre, Julia, Laptev, Ivan, Schmid, Cordelia, and Sivic, Josef. “Detecting Unseen Visual Relations Using Analogies”. In: *ICCV*. 2019, pp. 1981–1990.

[PY99] Pirrelli, Vito and Yvon, François. “Analogy in the lexicon: a probe into analogy-based machine learning of language”. In: *ISHC*. 1999.

[PR14] Prade, Henri and Richard, Gilles. “A Short Introduction to Computational Trends in Analogical Reasoning”. In: *CAAR-CT*. Vol. 548. SCI. Springer, 2014, pp. 1–22. ISBN: 978-3-642-54516-0. DOI: 10.1007/978-3-642-54516-0_1.

[PR18] Prade, Henri and Richard, Gilles. “Analogical proportions: From equality to inequality”. In: *IJAR* 101 (2018), pp. 234–254. DOI: 10.1016/j.ijar.2018.07.005.

[PR21] Prade, Henri and Richard, Gilles. “Analogical Proportions: Why They Are Useful in AI”. In: *IJCAI*. Survey Track. 2021, pp. 4568–4576. DOI: 10.24963/ijcai.2021/621.

[Pra+22] Pradhan, Sameer, Bonn, Julia, Myers, Skatje, Conger, Kathryn, O’gorman, Tim, Gung, James, Wright-bettner, Kristin, and Palmer, Martha. “PropBank Comes of Age—Larger, Smarter, and more Diverse”. In: **SEM*. ACL, 2022, pp. 278–288. DOI: 10.18653/v1/2022.starsem-1.24.

[Raj+16] Rajpurkar, Pranav, Zhang, Jian, Lopyrev, Konstantin, and Liang, Percy. “SQuAD: 100, 000+ Questions for Machine Comprehension of Text”. In: *EMNLP*. ACL, 2016, pp. 2383–2392. DOI: 10.18653/v1/d16-1264.

[Ree+15] Reed, Scott E., Zhang, Yi, Zhang, Yuting, and Lee, Honglak. “Deep Visual Analogy-Making”. In: *NIPS*. Curran Associates, Inc., 2015, pp. 1252–1260. URL: http://papers.nips.cc/paper/5845-deep-visual-analogy-making.pdf.

[Ric03] Richter, Michael M. “Knowledge Containers”. In: *Readings in CBR*. 2003. URL: %7Bhttps://www.researchgate.net/publication/225070310_Knowledge_Containers%7D.

[RH99] Ringen, Catherine O. and Heinämäki, Orvokki. “Variation in Finnish Vowel Harmony: An OT Account”. In: *Nat. Lang. Linguist. Theory* 17.2 (1999), pp. 303–337. ISSN: 0167806X, 15730859. URL: http://www.jstor.org/stable/4047991.

[RDL17] Rogers, Anna, Drozd, Aleksandr, and Li, Bofang. “The (too Many) Problems of Analogical Reasoning with Word Vectors”. In: *Joint Conference on Lexical and Computational Semantics*. ACL, 2017, pp. 135–148. DOI: 10.18653/v1/S17-1017.

[Rud16] Ruder, Sebastian. “An overview of gradient descent optimization algorithms”. In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[RA73] Rumelhart, David E. and Abrahamson, Adele A. “A model for analogical reasoning”. In: *CogPsy* 5.1 (1973), pp. 1–28.

[Rup+16] Ruppenhofer, Josef, Ellsworth, Michael, Petruck, Miriam R. L., Johnson, Christopher R., and Scheffczyk, Jan. *FrameNet II: Extended theory and practice*. 2016. URL: https://framenet2.icsi.berkeley.edu/docs/r1.7/book.pdf.

[SZF15] Sadeghi, Fereshteh, Zitnick, C. Lawrence, and Farhadi, Ali. “Visalogy: Answering Visual Analogy Questions”. In: *NIPS*. Curran Associates, Inc., 2015, pp. 1882–1890. URL: http://papers.nips.cc/paper/5777-visalogy-answering-visual-analogy-questions.pdf.

[San+19] Sanh, Victor, Debut, Lysandre, Chaumond, Julien, and Wolf, Thomas. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *CoRR* abs/1910.01108 (2019). arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108.

[Sar21] Sarker, Iqbal H. “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions”. In: *SN Comput. Sci.* 2.6 (2021), p. 420. DOI: 10.1007/S42979-021-00815-1.

[SK95] Smyth, Barry and Keane, Mark T. “Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems”. In: *IJCAI*. Morgan Kaufmann, 1995, pp. 377–382.

[SM01] Smyth, Barry and McKenna, Elizabeth. “Competence Models and the Maintenance Problem”. In: *Comput. Intell.* 17 (2001), pp. 235–249.

[Son+20] Song, Kaitao, Tan, Xu, Qin, Tao, Lu, Jianfeng, and Liu, Tie-Yan. “MPNet: Masked and Permuted Pre-training for Language Understanding”. In: *NIPS*. Vol. 33. Curran Associates, Inc., 2020, pp. 16857–16867. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/c3a690be93aa602ee2dc0ccab5b7b67e-Paper.pdf.

[SGS15] Srivastava, Rupesh K., Greff, Klaus, and Schmidhuber, Jürgen. “Training Very Deep Networks”. In: *NIPS*. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/215a71a12769b056c3c32e7299f1c5ed-Paper.pdf.

[SY04] Stroppa, Nicolas and Yvon, François. “Analogies dans les séquences : un solveur à états finis”. In: *Actes de la 11ème conférence sur le Traitement Automatique des Langues Naturelles. Posters, TALN 2004, Fès, Maroc, April 2004*. ATALA, 2004, pp. 131–140. URL: https://aclanthology.org/2004.jeptalnrecital-poster.22/.

[SY07] Stroppa, Nicolas and Yvon, François. *Formal Models of Analogical Proportions*. Technical report 2006-D008, Télécom Paris. 2007. URL: https://hal.science/hal-00145148.

[SS22] Sultan, Oren and Shahaf, Dafna. “Life is a Circus and We are the Clowns: Automatically Finding Analogies between Situations and Processes”. In: *EMNLP*. ACL, 2022, pp. 3547–3562. URL: https://aclanthology.org/2022.emnlp-main.232.

[Swa+17] Swayamdipta, Swabha, Thomson, Sam, Dyer, Chris, and Smith, Noah A. *Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold*. 2017. DOI: 10.48550/ARXIV.1706.09528. arXiv: 1706.09528.

[Syl+15] Sylak-Glassman, John, Kirov, Christo, Post, Matt, Que, Roger, and Yarowsky, David. "A Universal Feature Schema for Rich Morphological Annotation and Fine-Grained Cross-Lingual Part-of-Speech Tagging". In: *International Workshop on Systems and Frameworks for Computational Morphology.* 2015.

[TWL20] Taillandier, Valentin, Wang, Liyan, and Lepage, Yves. "Réseaux de neurones pour la résolution d'analogies entre phrases en traduction automatique par l'exemple". FR. In: *TALN.* Vol. 2. ATALA, 2020, pp. 108–121.

[Ush+21] Ushio, Asahi, Anke, Luis Espinosa, Schockaert, Steven, and Camacho-Collados, José. "BERT is to NLP what AlexNet is to CV: Can Pre-Trained Language Models Identify Analogies?" In: *ACL/IJCNLP.* ACL, 2021, pp. 3609–3624. DOI: 10.18653/V1/2021.ACL-LONG.280.

[VSD21] Vandenbussche, Pierre-Yves, Scerri, Tony, and Daniel Jr, Ron. "Word Sense Disambiguation with Transformer Models". In: *SemDeep-6.* 2021, pp. 7–12.

[Van20] Vania, Clara. "On understanding character-level models for representing morphology". PhD thesis. University of Edinburgh, 2020.

[Vas+17] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, and Polosukhin, Illia. "Attention is All you Need". In: *NIPS.* 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[WS10] Walther, Géraldine and Sagot, Benoît. "Developing a Large-Scale Lexicon for a Less-Resourced Language: General Methodology and Preliminary Experiments on Sorani Kurdish". In: 2010.

[WSF10] Walther, Géraldine, Sagot, Benoît, and Fort, Karën. "Fast Development of Basic NLP Tools: Towards a Lexicon and a POS Tagger for Kurmanji Kurdish". In: *LGC.* 2010. URL: http://web.me.com/gwalther/homepage/Publications_(fr)_files/clg10kmr.pdf.

[WL20] Wang, Liyan and Lepage, Yves. "Vector-to-Sequence Models for Sentence Analogies". In: *ICACSIS.* 2020, pp. 441–446. DOI: 10.1109/ICACSIS51025.2020.9263191.

[Wan+20a] Wang, Wenhui, Wei, Furu, Dong, Li, Bao, Hangbo, Yang, Nan, and Zhou, Ming. "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers". In: *NIPS.* 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[Wan+20b] Wang, Yaqing, Yao, Quanming, Kwok, James T., and Ni, Lionel M. "Generalizing from a Few Examples: A Survey on Few-shot Learning". In: *ACM Comput. Surv.* 53.3 (2020). ISSN: 0360-0300. DOI: 10.1145/3386252.

[WS22] Wang, Yutong and Scott, Clayton. "VC dimension of partially quantized neural networks in the overparametrized regime". In: *ICLR.* 2022. URL: https://openreview.net/forum?id=7udZAsEzd60.

[Wen18] Weng, Lilian. "Attention? Attention!" In: *lilianweng.github.io* (2018). URL: https://lilianweng.github.io/posts/2018-06-24-attention/.

[Wij+23] Wijesiriwardene, Thilini, Wickramarachchi, Ruwan, Gajera, Bimal, Gowaikar, Shreeyash, Gupta, Chandan, Chadha, Aman, Reganti, Aishwarya Naresh, Sheth, Amit, and Das, Amitava. "ANALOGICAL - A Novel Benchmark for Long Text Analogy Evaluation in Large Language Models". In: *ACL.* ACL, 2023, pp. 3534–3549. DOI: 10.18653/v1/2023.findings-acl.218.

[WL01] Wilson, D. and Leake, D. "Maintaining Case-Based Reasoners: Dimensions and Directions". In: *Comput. Intell.* 17.2 (2001), pp. 196–213.

[WM03] Wilson, D. Randall and Martinez, Tony R. "The general inefficiency of batch training for gradient descent learning". In: *NN* 16.10 (2003), pp. 1429–1451. DOI: 10.1016/S0893-6080(03)00138-2.

[Wu+16] Wu, Yonghui, Schuster, Mike, Chen, Zhifeng, Le, Quoc V., Norouzi, Mohammad, Macherey, Wolfgang, Krikun, Maxim, Cao, Yuan, Gao, Qin, Macherey, Klaus, Klingner, Jeff, Shah, Apurva, Johnson, Melvin, Liu, Xiaobing, Kaiser, Lukasz, Gouws, Stephan, Kato, Yoshikiyo, Kudo, Taku, Kazawa, Hideto, Stevens, Keith, Kurian, George, Patil, Nishant, Wang, Wei, Young, Cliff, Smith, Jason, Riesa, Jason, Rudnick, Alex, Vinyals, Oriol, Corrado, Greg, Hughes, Macduff, and Dean, Jeffrey. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016). arXiv: 1609.08144. URL: http://arxiv.org/abs/1609.08144.

[Yas+24] Yasunaga, Michihiro, Chen, Xinyun, Li, Yujia, Pasupat, Panupong, Leskovec, Jure, Liang, Percy, Chi, Ed H., and Zhou, Denny. *Large Language Models as Analogical Reasoners*. 2024. URL: https://openreview.net/forum?id=AgDICX1h50.

[Yvo03] Yvon, François. "Finite-state transducers solving analogies on words". In: *Rapport GET/ENST&LTCI* (2003).

[Zer23] Zervakis, Georgios. "Enriching large language models with semantic lexicons and analogies". Theses. Université de Lorraine, 2023. URL: https://theses.hal.science/tel-04138899.

[Zer+22] Zervakis, Georgios, Vincent, Emmanuel, Couceiro, Miguel, Schoenauer, Marc, and Marquer, Esteban. "An analogy based approach for solving target sense verification". In: *NLPIR*. 2022. URL: https://hal.inria.fr/hal-03792071.

[Zha+19] Zhang, Zhengyan, Han, Xu, Liu, Zhiyuan, Jiang, Xin, Sun, Maosong, and Liu, Qun. "ERNIE: Enhanced Language Representation with Informative Entities". In: *ACL*. ACL, 2019, pp. 1441–1451. DOI: 10.18653/v1/P19-1139.

[ZFL22] Zhou, Yifei, Fam, Rashel, and Lepage, Yves. "Extraction of analogies between sentences on the level of syntax using parse trees". In: *ATA@ICCBR*. Vol. 3389. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 30–42. URL: https://ceur-ws.org/Vol-3389/ICCBR%5C_2022%5C_Workshop%5C_paper%5C_72.pdf.

[ZM20] Zhu, Xunjie and Melo, Gerard de. "Sentence Analogies: Linguistic Regularities in Sentence Embeddings". In: *COLING*. ACL, 2020, pp. 3389–3400. DOI: 10.18653/v1/2020.coling-main.300.

Acronyms

KB Kilobyte.

MB Megabyte.

GB Gigabyte.

UTF8 Unicode Transformation Format – 8-bit.

GPU Graphics Processing Unit.

AP Analogical Proportion.
AE Auto-Encoder.

NN artificial Neural Network.

MLP Multi-Layer Perceptron.

ReLU Rectified Linear Unit.

GLU Gated Linear Unit.

CEL Cosine Embedding Loss.

MSE Mean Squared Error.
ML Machine Learning.

DL Deep Learning.

CNN Convolutional Neural Network.

GNN Graph Neural Network.

RNN Recurrent Neural Network.

LSTM Long- and Short-Term Memory neural network.

BiLSTM Bidirectionnal LSTM.
BiHLSTM Bidirectionnal Hierarchical Long- and Short-Term Memory neural network.
BERT Bidirectional Encoder Representations from Transformer.

mBert Multilingual BERT.

SGD Stochastic Gradient Decent.

NLTK Natural Language Toolkit.

TLFi Trésor de la langue Françaises informatisé.

CNTRL Centre National de Resources Textuelles et Lexicales.

NLP Natural Language Processing.
Nlg Lepage's Nlg toolkit.

cbow Continuous Bag of Words.

GloVe Global Vectors.

LSA Latent Semantic Analysis.

SIGMORPHON ACL Special Interest Group on Computational Morphology and Phonology.
Sig16 Sigmorphon 2016 Task 1.
Sig19 Sigmorphon 2019 Task 1.
Sig23 Sigmorphon 2023 Task 0.

JBATS Japanese Bigger Analogy Test Set.

BATS Bigger Analogy Test Set.

Kakenhi-Sig16 Kakenhi 15K00317 word analogies from Sigmorphon 2016 Task 1.

ANN framework Analogy Neural Network framework.
ANNc Analogy Neural Network for classification.
ANNr Analogy Neural Network for retrieval/generation.
ANNa Analogy Neural Network web application.

CNN-emb CNN-based word embedding.

AE-emb AE-based word embedding.

CNN+3CosAdd 3CosAdd combined with CNN-emb.
CNN+3CosMul 3CosMul combined with CNN-emb.
CNN+ANNc ANNc combined with CNN-emb.
CNN+ANNr ANNr combined with CNN-emb.

AE+ANNr ANNr combined with AE-emb.

AE+par. parallelogram rule combined with AE-emb.

CWA Closed World Assumption.

OWA Open World Assumption.

POS Part Of Speech.

SMT Structure Mapping Theory.

BOW beginning of word.

EOW end of word.

VC dimension Vapnik-Chervonenkis dimension.

FE Frame Element.

core FE core Frame Element.

non-core FE peripheral/extra-thematic Frame Element.

SR Semantic Role.

FSRL Frame Semantic Role Labeling.
FN FrameNet.
FN1.7 FrameNet-1.7.
FN1.5 FrameNet-1.5.

SotA State of the Art.

Ex-QA Extractive Question Answering.

QA-SRL Question-Answer Driven Semantic Role Labeling.

SQuAD Stanford Question Answering Dataset.

WER Word Error Rate.

SPL Shortest Path Length.

PLM Pretrained Language Model.

WSD Word Sense Desambiguation.

TSV Target Sense Verification.

WiC-TSV Words-in-Context-TSV.

CBR Case-Based Reasoning.
CoAT Complexity Measure for Analogical Transfer.
CoAT-APC Complexity Measure for Analogical Transfer-Analogical Proportion based Classification.

MeATCube Measure of the complexity of a dataset for Analogical Transfer using slices of Boolean Cubes.

***k*-NN** *k*-Nearest Neighbors.

MCE Minimum Classification Error loss.

Abbreviations

w.r.t. Abbreviation of the English “with regards to”.

i.e. Abbreviation of the Latin *id est* meaning “that is”. From https://en.wiktionary.org/wiki/i.e..

e.g. Abbreviation of the Latin *exemplīgrātiā* meaning “for the sake of an example”. From https://en.wiktionary.org/wiki/e.g..

etc. Abbreviation of the Latin *et cetera* meaning “and the rest [of the things]; and the other things”. From https://en.wiktionary.org/wiki/etc..

et al. Abbreviation of the Latin *et aliī* meaning “and others”. From https://en.wiktionary.org/wiki/et_al..

Index