UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Neural Methods for Sentiment Analysis and Text Summarization

THÈSE

présentée et soutenue publiquement le 29 May 2020

pour l'obtention du

Doctorat de l'Université de Lorraine

(Mention Informatique)

par

Thien-Hoa Le

Composition du jury

Directeur :	Christophe Cerisara	Chargé de recherche 1, CNRS, LORIA, France
Rapporteurs :	Alexis Nasr	Professeur, Université d'Aix-Marseille, France
	Laurent Besacier	Professeur, Université Joseph Fourier de Grenoble, France
Examinateurs :	Claire Gardent	Directrice de recherche, CNRS, LORIA, France
	Enrique Elfonseca	Chercheur, Google, Zurich, Switzerland
	Irina Illina	Maître de conférence, Université de Lorraine, Nancy, France

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

Acknowledgements

The past 3 years and a half of my PhD was the most unforgettable and inspiring period so far in my life. In this long journey, though trying, I could not escape from the common painful, ups and downs patterns like anyone else. At the same time, I feel that I did not only grow up in the research thinking, but also in the human being side. Leaving academic research and going to private, I will not only miss its freedom environment, but I will also miss my advisors, my colleagues, my friends that always presented by my side during the bad and the good days.

First, I would like to express my deepest gratitude to my advisors, Christophe Cérisara and Claire Gardent. Christophe has been like a teacher, a coach and also a friend to me. He has always been at his position to support and to help me for almost any matters including personal ones. Thanks to him, I can experience a large range of activities of researchers: giving talks, teaching, supervising, mentoring master students, reviewing papers... I especially learned from him to be always humble. Being humble, our mind and our heart will be more open to great ideas from the others. When I proposed new, unexpected activities like doing internship in big tech company, launching a youtube channel on algorithms / data structures, participating in french-american seminar,... he always encouraged me. While letting me to be myself, he did not hesitate to tell me when to say no to the distractors. He is the best coach that I wished to have. I always feel pleasant to talk to him and no passing days I stop feeling being fortunate to have him as my advisor. Besides, I also would like to extend my deep gratitude to Claire, my unofficial co-advisor. Working along with her, I can feel a very young, profoundly kind and energetic soul of renowned researcher. She is always dynamic and presents actively in the research community while still be able to keep a good work-life balance. I learned how to become an independent researcher from her; working and collaborating with her is a real honour for me. She is a perfect role model that I want to head to. Without her help and her support, it will be uncertain if I can go all the way through my PhD.

I would also like to thank all the members of my thesis committee: Enrique Elfonseca, Alexis Nasr, Laurent Besacier, Irina Illina for providing feedback to my thesis. I am also thankful to Enrique Elfonseca, Daniele Pighin, Aliaksei Severyn, Massimo Nicosia, Katja Filippova, Sascha Rothe, and other researchers at Google Zurich who have shared me a lot of precious deep learning knowledge and research experiences. Thanks to this valuable practical internship, I could start my new direction of research on text summarization when I returned back to my lab and I could also teach Master student better on these matters. I am very grateful especially to Enrique Elfonseca for recommending me to his team for the internship and for serving as member in my jury thesis. He made an important turn to my life !

I am also thankful to my colleagues and engineers, researchers in Synalp team and in laboratory Loria (in alphabetical order): Alexandre Denis, Anastasia Shimorina, Angela Fan, Bikash Gyawali, Denis Paperno, Émilie Colin, Emmanuel Vincent, Ismael Bada, Joel Legrand, Laura Perez-Beltrachini, Timothée Mickus, Timothy Garwood, Yannick Parmentier. Thank you for all the supports, advices and for sharing your own experiences to help me avoid common pitfalls of PhD. Confiding with you helped me feel less painful when I was weakest while funny moments passing together made me stronger and believed more that we can all reach the end of these research apprentice's years. If advisors are the coach, you are my secret sources of nutrition to fight against any difficulties during the training.

Last but not least, I would like to thank my benefactors in France. The cost living in France is high, if there's no countless help, great expectation and much trust from

M. Rodolphe du Gardin, Mme. Thérèse Hudina and my uncle Jean-Baptiste Le, I could not feel reassured to finish my Master and my PhD. Certainly, my life is also meaningless without my family and my loved one. They always encourage and provide me support whenever I need. Especially, my father, my uncle and my big brothers played an important role in educating me and helped me pursue Computer Science throughout my studies. Hang was there at the end of my thesis when I need her the most, to support me to complete this wonderful journey of PhD. They are all my beloved person, I always wish that I will never let them down. I hope that this PhD will not be the last achievement but it will be just the start of many great and happy things coming that I can give them.

Résumé

Méthodes neuronales
pour l'analyse des sentiments et la synthèse des textes

Thien-Hoa Le

Cette thèse aborde deux questions majeures du traitement automatique du langage naturel liées à l'analyse sémantique des textes: la détection des sentiments, et le résumé automatique. Dans ces deux applications, la nécessité d'analyser le sens du texte de manière précise est primordiale, d'une part pour identifier le sentiment exprimé au travers des mots, et d'autre part pour extraire les informations saillantes d'une phrase complexe et les réécrire de la manière la plus naturelle possible tout en respectant la sémantique du texte d'origine. Nous abordons ces deux questions par des approches d'apprentissage profond, qui permettent d'exploiter au mieux les données, en particulier lorsqu'elles sont disponibles en grande quantité.

Analyse des sentiments neuronale. De nombreux réseaux de neurones convolutionnels profonds ont été adaptés du domaine de la vision aux tâches d'analyse des sentiments et de classification des textes. Cependant, ces études ne permettent pas de conclure de manière satisfaisante quant à l'importance de la profondeur du réseau pour obtenir les meilleures performances en classification de textes. Dans cette thèse, nous apportons de nouveaux éléments pour répondre à cette question. Nous proposons une adaptation du réseau convolutionnel profond DenseNet pour la classification de texte et étudions l'importance de la profondeur avec différents niveaux de granularité en entrée (mots ou caractères). Nous montrons que si les modèles profonds offrent de meilleures performances que les réseaux peu profonds lorsque le texte est représenté par une séquence de caractères, ce n'est pas le cas avec des mots. En outre, nous proposons de modéliser conjointement sentiments et actes de dialogue, qui constituent un facteur explicatif influent pour l'analyse du sentiment. Nous avons annoté manuellement les dialogues et les sentiments sur un corpus de micro-blogs, et entraîné un réseau multi-tâches sur ce corpus. Nous montrons que l'apprentissage par transfert peut être efficacement réalisé entre les deux tâches et analysons de plus certaines corrélations spécifiques entre ces deux aspects.

Résumé de texte neuronal. L'analyse de sentiments n'apporte qu'une partie de l'information sémantique contenue dans les textes et est insuffisante pour bien comprendre le texte d'origine et prendre des décisions fondées. L'utilisateur d'un tel système a également besoin des raisons sous-jacentes pour vraiment comprendre les documents. Dans cette partie, notre objectif est d'étudier une autre forme d'information sémantique fournie par les modèles de résumé automatique. Nous proposons ainsi un modèle de résumé qui présente de meilleures propriétés d'explicabilité et qui est suffisamment souple pour prendre en charge divers modules d'analyse syntaxique. Plus spécifiquement, nous linéarisons l'arbre syntaxique sous la forme de segments de texte superposés, qui sont ensuite sélectionnés par un apprentissage par renforcement (RL) et re-générés sous une forme compressée. Par conséquent, le modèle proposé est capable de gérer à la fois le résumé par extraction et par abstraction. En outre, les modèles de résumé automatique faisant de plus en plus appel à des approches d'apprentissage par renforcement, nous proposons une étude basée sur l'analyse syntaxique des phrases pour tenter de mieux comprendre quels types d'information sont pris en compte dans ces approches. Nous comparons ainsi de manière détaillée les modèles avec apprentissage par renforcement et les modèles exploitant une connaissance syntaxique supplémentaire des phrases ainsi que

leur combinaison, selon plusieurs dimensions liées à la qualité perçue des résumés générés. Nous montrons lorsqu'il existe une contrainte de ressources (calcul et mémoire) qu'il est préférable de n'utiliser que l'apprentissage par renforcement, qui donne des résultats presque aussi satisfaisants que des modèles syntaxiques, avec moins de paramètres et une convergence plus rapide.

Abstract

Neural Methods
for Sentiment Analysis and Text Summarization

Thien-Hoa Le

This thesis focuses on two Natural Language Processing tasks that require to extract semantic information from raw texts: Sentiment Analysis and Text Summarization. This dissertation discusses issues and seeks to improve neural models on both tasks, which have become the dominant paradigm in the past several years. Accordingly, this dissertation is composed of two parts: the first part (Neural Sentiment Analysis) deals with the computational study of people's opinions, sentiments, and the second part (Neural Text Summarization) tries to extract salient information from a complex sentence and rewrites it in a human-readable form.

Neural Sentiment Analysis. Similar to computer vision, numerous deep convolutional neural networks have been adapted to sentiment analysis and text classification tasks. However, unlike the image domain, these studies are carried on different input data types and on different datasets, which makes it hard to know if a deep network is truly needed. In this thesis, we seek to find elements to address this question, i.e. whether neural networks must compute deep hierarchies of features for textual data in the same way as they do in vision. We thus propose a new adaptation of the deepest convolutional architecture (DenseNet) for text classification and study the importance of depth in convolutional models with different atom-levels (word or character) of input. We show that deep models indeed give better performances than shallow networks when the text input is represented as a sequence of characters. However, a simple shallow-and-wide network outperforms the deep DenseNet models with word inputs. Besides, to further improve sentiment classifiers and contextualize them, we propose to model them jointly with dialog acts, which are a factor of explanation and correlate with sentiments but are nevertheless often ignored. We have manually annotated both dialogues and sentiments on a Twitter-like social medium, and train a multi-task hierarchical recurrent network on joint sentiment and dialog act recognition. We show that transfer learning may be efficiently achieved between both tasks, and further analyze some specific correlations between sentiments and dialogues on social media.

Neural Text Summarization. Detecting sentiments and opinions from large digital documents does not always enable users of such systems to take informed decisions, as other important semantic information is missing. People also need the main arguments and supporting reasons from the source documents to truly understand and interpret the document. To capture such information, we aim at making the neural text summarization models more explainable. We propose a model that has better explainability properties and is flexible enough to support various shallow syntactic parsing modules. More specifically, we linearize the syntactic tree into the form of overlapping text segments, which are then selected with reinforcement learning (RL) and regenerated into a compressed form. Hence, the proposed model is able to handle both extractive and abstractive summarization. Further, we observe that RL-based models are becoming increasingly ubiquitous for many text summarization tasks. We are interested in better understanding what types of information is taken into account by such models, and we propose to study this question from the syntactic perspective. We thus provide a detailed comparison of both RL-based and syntax-aware approaches and of their combination along several dimensions that

relate to the perceived quality of the generated summaries such as number of repetitions, sentence length, distribution of part-of-speech tags, relevance and grammaticality. We show that when there is a resource constraint (computation and memory), it is wise to only train models with RL and without any syntactic information, as they provide nearly as good results as syntax-aware models with less parameters and faster training convergence.

"Imagine two worlds, one with you and one without you. What's the difference between the two worlds? Maximize that difference. That's the meaning of your life."

Kai-Fu Lee, Making a World of Difference

Contents

Résumé en français

Depuis quelques années, la quantité de données disponibles augmente de façon exponentielle. Plus de données offrent plus d'opportunités, mais de nombreux problèmes nouveaux et de grands défis se posent également. On estime que 90% des données humaines ont été générées au cours des deux dernières années. Si on ne considère que les données textuelles, 2 milliards d'utilisateurs actifs de Facebook mettent à jour environ 734 millions de statuts et commentaires par jour. Google doit maintenant traiter plus de 3,5 milliards de recherches par jour. Environ 23 milliards de SMS et 225 milliards de courriels électroniques sont envoyés chaque jour pour communiquer au travail et à des fins personnelles [1]. Cette quantité, cette diversité et cette rapidité de diffusion des données sont sans précédent. Compte tenu du rythme croissant des données entrantes, du temps et de la capacité humaine d'analyse des informations limités, il existe un besoin urgent de créer des outils automatiques pour aider les hommes à gérer et exploiter ces données.

L'analyse des sentiments exprimés dans les textes est un outil essentiel de nos jours, qui permet aux entreprises et aux gouvernements de mieux comprendre la manière dont sont perçus par les utilisateurs et les citoyens de nouveaux produits ou de nouvelles propositions. À l'ère des échanges sur le marché mondial impliquant des milliers de concurrents issus de cultures différentes, l'analyse des sentiments est l'application de Traitement Automatique du Langage Naturel (TALN) la plus populaire pour les entreprises. Les gouvernements modernes ont également besoin de cet outil pour adapter plus rapidement leur politique à la réaction en temps réel de leurs citoyens sur les médias sociaux. En ce qui concerne les individus, leur vision de voir être créé à long terme un assistant, une machine au comportement à la fois humain et convivial, capable de communiquer et d'aider l'homme dans la vie quotidienne (robots en tant que réceptionniste, assistants, ...) restera un rêve tant que les aspects sémantiques essentiels du langage naturel ne seront pas interprétés correctement par nos modèles. De plus, alors que le temps est devenu un facteur rare de nos jours, lire toutes les informations textuelles disponibles n'est pas réaliste et les besoins d'accéder à des résumés fidèles et naturels de tous ces documents, avec les arguments principaux exprimant les idées centrales clairement identifiés, est devenu un impératif que ne peuvent réaliser les méthodes traditionnelles basées sur les mots fréquents.

Dans ce travail de thèse, mon objectif principal est de faire progresser les approches neuronales pour l'analyse des sentiments (Paragraphe 1) et pour le résumé automatique des phrases (Paragraphe 2). Je m'inspire fortement pour cela des nouvelles approches développées au cours des cinq dernières années avec l'essor de l'apprentissage profond (Goodfellow, Bengio, and Courville, 2017).

[1] Selon l'article de Forbes: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/.

1. Analyse des sentiments

L'analyse neuronale des sentiments exprimés dans les textes consiste en une architecture de réseau neuronal à plusieurs couches qui permet de bien exploiter la représentation des données lorsque le réseau devient de plus en plus grand. Pour faire face à la grande dimensionnalité de l'approche traditionnelle du sac de mots (*Bag-of-Words*), l'analyse de sentiment neuronale utilise les plongements vectoriels des mots (*word embedding*) (Bengio, Ducharme, and Vincent, 2000; Collobert et al., 2011), qui sont des vecteurs denses de faible dimension servant à coder la sémantique et les propriétés syntaxiques lexicales. Dans l'espace des plongements de mots, les vecteurs caractéristiques (*feature vectors*) des mots similaires se retrouvent au même endroit; la transformation syntaxique de *"roi"* et de *"reine"* en *"homme"* et *"femme"* peut être exprimée de manière presque équivalente par la relation linéaire entre leurs vecteurs respectifs (Mikolov et al., 2013). Un deuxième attrait important des méthodes d'apprentissage profond pour résoudre ces problèmes relève de leur capacité à découvrir et exploiter les signaux faibles présents dans de grandes quantités de données, ce qui leur permet de mieux généraliser et de dépasser les performances des méthodes traditionnelles d'apprentissage automatique.

Cependant, contrairement aux tâches de classification des images ou de la parole, il est encore difficile aujourd'hui de trouver un consensus de la communauté de recherche sur la meilleure architecture possible de réseau neuronal pour l'analyse des sentiments. Au début, une école de pensée stipulait que des informations syntaxiques et sémantiques explicites, telles que des arbres syntaxiques, des lexiques d'opinion, des parties du discours,.. etc peuvent aider à mieux composer les sentiments des mots d'une phrase. Ainsi, Socher et al., (2011) ont d'abord introduit un réseau semi-supervisé d'auto-encodeurs récursifs (RAE) pour obtenir une représentation vectorielle pour la classification du sentiment de la phrase. Ensuite, Socher et al., (2012) ont proposé un réseau de tenseur neuronal récursif (RNTN) afin de prédire plus précisément les effets sémantiques de la composition des mots à l'aide d'un arbre d'analyse de phrases entièrement étiqueté obtenu à partir d'un analyseur externe. Qian et al., (2015) se sont alors rendus compte qu'une composition des plongements d'étiquettes renforcerait les plongements de mots, ils ont donc proposé deux modèles: un réseau de neurones récursif guidé par les étiquettes (TG-RNN) et un réseau de neurones récursif étiqueté (TE-RNN). Ces approches ont d'abord montré de nombreux avantages potentiels des réseaux de neurones pour l'analyse des sentiments, mais accompagnés du désavantage de devoir calculer des arbres syntaxiques pour extraire les caractéristiques des phrases. Grâce à une conception particulièrement soignée, un modèle de réseau de neurones convolutionnel (CNN) ou un réseau de neurones récurrent (RNN) simple et efficace s'est avéré plus tard pouvoir également apprendre des relations intrinsèques entre les mots. Ces modèles sont attrayants car ils peuvent être entraînés de bout-en-bout directement à partir de l'objectif d'apprentissage, ce qui permet d'éliminer des composants inutiles et une meilleure exploitation des données d'entrée. Ainsi, Kalchbrenner, Grefenstette, and Blunsom, (2014) ont introduit un CNN dynamique (DCNN) qui utilise un opérateur de k-max pooling dynamique en tant que fonction de sous-échantillonnage non linéaire pour la modélisation sémantique des phrases. Kim, (2014) a expérimenté plusieurs variantes de CNN-rand, CNN-statique, CNN-non-statique pour la classification de sentiment des phrases. Wang, Jiang, and Luo, (2016) ont proposé une architecture conjointe CNN et RNN pour profiter des dépendances à longue distance apprises via le RNN et des entités locales capturées par le CNN.

De plus, bien qu'il soit populaire, le modèle neuronal n'est pas encore bien étudié

pour l'analyse des sentiments dans le contexte du dialogue, alors qu'il existe des relations étroites et de fortes influences respectives entre ces deux aspects. Intuitivement, considérons un dialogue dans lequel l'utilisateur A écrit quelque chose de *positif* (sentiment) concernant une marque de téléphone intelligent donnée. Ensuite, l'utilisateur B *est en désaccord* (acte de dialogue) et indique un aspect *négatif* de cette marque. Une corrélation évidente entre actes de dialogue et sentiments se retrouve dans cet exemple trivial. D'autre part, l'une des tâches de TALN les plus étudiées sur Twitter est la reconnaissance des sentiments, qui est organisée chaque année à la campagne d'évaluation de SEMEVAL (Nakov et al., 2016b; Bethard et al., 2017). Inversement, il y a moins de travaux qui étudient les dialogues sur Twitter. Les dialogues sur les médias sociaux ont une forme très différente de celle des autres médias, tels que la voix, les SMS et les réunions, car les participants au dialogue sont souvent inconnus à l'avance, ils peuvent venir et partir à tout moment et vivre dans des fuseaux horaires différents. La structure des dialogues sur les médias sociaux peut donc être un arbre, ou un graphe orienté acyclique si nous prenons en compte @-mentions. Cela rend la modélisation plus difficile que dans le cas des conversations en direct traditionnelles.

Dans la suite, nous voulons tout d'abord proposer des éléments factuels sur la nécessité d'appliquer une structure de réseau profonde aux tâches d'analyse des sentiments et de classification de texte (Paragraphe 1.1). Si l'utilité des grandes profondeurs de réseau a été clairement confirmée dans le domaine de la vision par ordinateur, les recherches sur l'analyse des sentiments montrent des divergences à ce sujet. En outre, la majorité des travaux sur les systèmes d'analyse de sentiments utilisent uniquement des données textuelles brutes, hors contextes, en entrée. De telles approches ne sont clairement pas suffisantes pour modéliser les opinions et les comportements humains dans conditions réelles, en contexte. De plus, les actes de dialogue dans les conversations ne sont toujours pas explorés de manière approfondie pour les tâches d'analyse des sentiments. Nous souhaitons donc pallier ces déficiences en proposant un modèle joint, intégrant l'analyse de sentiments dans des situations réelles et prenant en compte les interactions entre les participants d'un dialogue (Paragraphe 1.2).

1.1 Impact de la profondeur du réseau de neurones pour l'analyse des sentiments

Suite au succès des approches d'apprentissage profond dans le concours ImageNet, l'application de modèles profonds pour la classification de texte et l'analyse des sentiments a suscité beaucoup d'intérêt, des réseaux de plus en plus profonds étant proposés dans la littérature: (Kim, 2014) (couche CNN peu profonde et large), (Zhang, Zhao, and LeCun, 2015) (6 couches CNN), (Schwenk et al., 2017) (29 couches CNN). En parallèle du développement des réseaux profonds, un autre débat existe sur le type d'unités (mot ou caractère) qui serait le plus efficace pour les tâches de TALN. Tout d'abord, les plongements des mots, qui sont des représentations continues de mots, initialement proposés par Bengio, Ducharme, and Vincent, (2000) et largement adoptés après word2vec (Mikolov et al., 2013), ont été choisis comme les principales représentations pour la plupart des tâches. Sur la base de telle représentation, il est généralement admis que, de la même manière que pour la vision, le modèle apprendra des caractéristiques hiérarchiques du texte: les mots se combinent pour former des n-grammes, des phrases, des séquences... Néanmoins, il n'est pas encore certain que les réseaux très profonds basés sur des mots soient vraiment les meilleurs pour

la classification de texte (Santos and Gatti, 2014; Kim et al., 2016; Dhingra et al., 2017; Yang, Salakhutdinov, and Cohen, 2016).

Pour apporter de nouvelles idées ou confirmer les hypothèses précédentes, nous proposons l'adaptation de DenseNet (Huang et al., 2016) (au Chapitre 3) pour la classification du texte et l'analyse des sentiments. Nous apportons des éléments de réponse en fournissant une comparaison complète avec un CNN peu profond et large (Kim, 2014) basé sur des caractères et des mots sur les cinq jeux de données décrits dans Zhang, Zhao, and LeCun, (2015). Ces expériences nous permettent de comparer l'influence croisée des unités utilisées en entrée (mots ou caractères) et de la typologie des réseaux (profonds ou peu profonds). En évaluant plusieurs tâches de classification de texte et d'analyse des sentiments, nous montrons qu'un réseau de neurones convolutif peu profond sur des mots reste le plus efficace, et qu'augmenter la profondeur de tels modèles convolutifs avec des mots en entrée n'apporte pas d'amélioration importante. Inversement, les modèles profonds surpassent les réseaux peu profonds lorsque le texte d'entrée est codé sous forme d'une séquence de caractères, mais même si ces modèles profonds approchent des performances des réseaux sur les mots, ils leur restent encore inférieurs en moyenne.

1.2 Analyse de sentiment et acte de dialogue

Comme le montre Pluwak, (2016), une analyse de sentiments performante ne peut se limiter aux seuls aspects lexicaux, c'est-à-dire à la combinaison de mots ayant une polarité explicite, mais doit également prendre en compte l'expression de sentiments implicite dans les textes, qui a été détectée dans près de la moitié des textes analysés (Pluwak, 2016). Il ne s'agit donc pas d'un épi-phénomène, mais bien d'un autre mode d'expression des sentiments, qu'il convient d'investiguer par d'autres moyens que les seuls modèles lexicaux. La théorie des actes de dialogue (ou plus généralement *speech acts*) constitue une approche pertinente pour aborder ce nouveau champ d'étude de l'analyse des sentiments.

Pour donner un exemple dans le domaine du sport: *"Ramenez l'ancien gardien de but!"*, ne communique pas explicitement un sentiment, mais exprime un acte illocutoire de *requête / demande / conseil*, duquel il est possible d'inférer : 1) le mécontentement du public envers le joueur actuel et 2) la préférence pour le joueur précédent. Les deux sont des indicateurs implicites, car faisant intervenir des concepts issus d'un modèle du monde, d'un jugement négatif envers ce gardien de but. L'utilisation d'une analyse de la fonction illocutoire des énoncés est donc indispensable, bien que non suffisante, pour interpréter correctement ces énoncés. Mais même sans modèle du monde sous-jacent, les interactions entre actes de dialogue et sentiments sont nombreux, par exemple lorsqu'une expression de désaccord ("*Non ! je ne pense pas comme toi [que ce joueur est mauvais]*") transforme un sentiment négatif en son contraire. Nous nous intéressons donc à ces interactions entre actes de dialogue et sentiments, dans la lignée d'autres chercheurs du domaine. Ainsi, Clavel and Callejas, (2016) ont étudié l'impact de la reconnaissance des dialogues et de l'analyse des sentiments sur les plateformes conversationnelles homme-agent et les interfaces conversationnelles affectives. Un agent conversationnel virtuel interagissant avec les humains, doit tenir compte des comportements et des attitudes émotionnels humains afin d'adapter son comportement en conséquence. En outre, la détection des émotions peut servir à aider le classifieur d'actes de dialogue. Boyer et al., (2011) ont exploité les expressions émotionnelles visuelles sur les visages pour mieux classer les actes de dialogue. Novielli and Strapparava, (2013) ont regroupé

des lexiques d'actes de dialogue et étudié la charge émotionnelle de ces actes de dialogue. Les auteurs ont également tenté d'améliorer ce classifieur en exploitant les lexiques affectifs avec des résultats encourageants. Des actes de dialogue peuvent également faciliter la tâche de détection des émotions. Hasegawa et al., (2013) ont montré comment prévoir les émotions dans les dialogues en ligne. Ils ont considéré une histoire qui comprend un énoncé et une réponse et ont entraîné un système pour prévoir l'émotion du destinataire provoquée par la réponse. Dans un esprit similaire, Herzig et al., (2016) ont considéré le contexte complet d'un dialogue plus long et ont inclus des caractéristiques de dialogue (par exemple, le temps écoulé entre les tours ...) dans un classifieur d'émotions.

À notre connaissance, le premier véritable modèle joint de sentiments et d'actes de dialogue est d'abord proposé dans Kim and Kim, (2018). Les auteurs ont exploité un réseau de convolution par tâche, dont les vecteurs de sortie sont enchaînés et transmis à un classifieur par tâche. La fusion des tâches repose sur de fortes hypothèses simplificatrices, notamment:

- L'acte de dialogue à l'instant t ne dépend pas de sentiments, mais dépend de l'acte de dialogue à l'instant $t-1$;
- Le sentiment au temps t ne dépend que du sentiment et de l'acte de dialogue au temps t (pas d'hypothèse de Markov);

Avec ces hypothèses, les auteurs obtiennent de meilleurs résultats lorsqu'ils reconnaissent conjointement des actes de dialogue et des sentiments sur un corpus coréen. Cependant, leur corpus et leur code ne sont pas distribués, ce qui rend les évaluations comparatives avec ce modèle difficile. Par rapport à ce travail, notre réseau proposé (au Chapitre 4) modélise conjointement les deux tâches avec des réseaux plus profonds, après la couche du plongement des mots. De plus, nos hypothèses sont plus faibles, car nous supposons que les deux étiquettes au temps t dépendent des deux tâches aux temps t et $t-1$, et la récurrence est appliquée aux niveaux des mot et des dialogues.

Nous montrons également que l'apprentissage par transfert est efficace lorsque le nombre d'étiquettes annotées pour une tâche est plus petit que pour l'autre tâche. Nous avons ensuite analysé la corrélation entre les deux tâches et montré que, bien qu'il y ait assez d'informations mutuelles pour permettre l'apprentissage par transfert, les deux tâches ne sont pas fortement corrélées globalement, et abordent chacune un niveau sémantique / pragmatique différencié. Enfin, bien que ces deux tâches soient caractérisées par des dynamiques différentes, nous avons identifié quelques motifs spécifiques présentant une forte corrélation. Toutes ces études sont réalisées sur un nouveau corpus extrait du réseau social Mastodon, qui facilite la conception d'expériences reproductibles. Le corpus annoté et le code source de notre modèle sont disponibles sous une licence ouverte et gratuite sur github.

2. Résumé automatique

Extraire l'information des sentiments à partir de documents numériques sur Internet permet aux lecteurs de saisir rapidement certaines des informations pertinentes qui les intéressent. Cependant, même lorsque cela est fait au niveau de l'aspect et du sujet, il reste encore beaucoup d'informations sémantiques utiles qui sont contenues dans les textes et qui ne sont pas facilement accessibles aux lecteurs sans une lecture attentive du document complet. Pour des applications pratiques, la tâche d'analyse des sentiments est donc rarement suffisante à elle seule et il faut envisager

d'autres méthodes d'analyse de textes pour aider les personnes à comprendre le contenu important sans devoir lire en détail l'ensemble des documents. Le résumé automatique de texte, c'est-à-dire la réduction d'un texte à son contenu essentiel, est donc une tâche intéressante qui complète l'analyse de sentiments en répondant aux questions *"Quoi"*, *"Pourquoi"* et *"Comment"* . Avec la vitesse croissante actuelle d'accumulation des données, le résumé automatique permettrait ainsi d'accéder à plus de contenus en moins de temps, d'aider à accélérer la recherche d'un document et à faciliter la communication avec les utilisateurs via un assistant automatique tel que Google Home, Amazon Alexa, ...

En fonction du type de sortie attendu, le résumé de texte est classé en méthodes d'extraction et d'abstraction. L'extraction est la réunion de fragments contenant des informations essentielles d'un document source. L'extraction est la méthode la plus facile à mettre en oeuvre et peut également servir d'aperçu du contenu du texte original. L'abstraction cherche à produire des résumés qui ne sont pas simplement une copie de certaines phrases du texte d'entrée, en reformulant et en fusionnant des idées et souvent en les exprimant avec d'autres termes. Les textes ainsi abstraits semblent plus naturels et imitent mieux le comportement humain, mais sont également très difficiles à produire.

Au cours des dernières années, grâce à des progrès importants réalisés dans le développement des méthodes neuronales, la majorité des approches de résumé automatique est progressivement passée des techniques d'extraction aux techniques abstractives. Dans les méthodes d'abstraction classiques, chaque composant est réglé séparément, par exemple l'extraction d'informations, la sélection du contenu et la réalisation de surface, ce qui rend difficile leur combinaison et l'amélioration globale du système. De ce fait, la qualité du résumé a stagné et des modifications majeures des systèmes existants étaient effectivement nécessaires. Le résumé de texte neuronal est une nouvelle approche qui résout les problèmes susmentionnés. Le résumé de texte neuronal ne nécessite qu'une connaissance minimale du domaine, mais s'appuie le plus souvent sur un corpus parallèle de paires de documents source - résumé, et il peut être entraîné de bout-en-bout en optimisant directement l'objectif d'apprentissage sans avoir à apprendre plusieurs composants différents. Le travail de Rush, Chopra, and Weston, (2015) sur l'application de la traduction automatique neurale au résumé est le premier à susciter un nouveau moyen de construire des systèmes de résumé abstractifs. Depuis, ce modèle neuronal séquence-à-séquence (Sutskever, Vinyals, and Le, 2014) est devenu la technologie de base de la plupart des systèmes abstractifs modernes.

Les premières mises en oeuvre d'approches neuronales à des tâches de génération de texte telles que la traduction automatique, le résumé et le sous-titrage des images font principalement appel à un apprentissage optimisant l'entropie croisée par simple descente de gradient stochastique. Des travaux plus récents montrent toutefois que l'apprentissage par renforcement (RL) ou l'enrichissement explicite des caractéristiques linguistiques supplémentaires (Sennrich and Haddow, 2016; Nallapati et al., 2016; Li et al., 2017; Le et al., 2017; Kiperwasser and Ballesteros, 2018) contribue à l'amélioration des performances. L'apprentissage par renforcement (RL) est notamment proposé pour remédier aux problèmes liés à l'apprentissage classique. Premièrement, il existe un décalage entre la façon dont le modèle est entraîné (conditionné par les vrais labels) et utilisé au moment du test (avec un argmax ou une recherche en faisceau), dénommé le problème du *biais d'exposition*. Deuxièmement, les métriques d'évaluation (par exemple ROUGE, METEOR, BLEU, etc.) diffèrent de l'objectif maximisé avec l'entropie croisée standard sur chaque jeton ; c'est ce qu'on appelle le problème de *non-concordance des pertes*. L'apprentissage par renforcement

est généralement utilisé pour optimiser des objectifs spécifiques à une tâche, tels que ROUGE pour les systèmes de résumé automatique (Paulus, Xiong, and Socher, 2018; Narayan, Cohen, and Lapata, 2018; Celikyilmaz et al., 2018; Pasunuru and Bansal, 2018) et SARI (Xu et al., 2016) pour les modèles de simplification de phrases (Zhang and Lapata, 2017).

Le modèle de résumé de texte séquence-à-séquence relance l'espoir de construire les résumés abstractifs longtemps attendus. Cependant, ce souhait est encore hors de notre portée, car de nombreux problèmes subsistent : les systèmes actuels gèrent de manière insatisfaisante les longues séquences (Chen and Bansal, 2018), génèrent souvent des hallucinations et des faits erronés (Amplayo, Lim, and Hwang, 2018; Cao et al., 2017; Song, Zhao, and Liu, 2018; Li et al., 2018a; Li et al., 2018b), etc. Nous proposons un modèle qui présente de meilleures propriétés d'explicabilité et qui est suffisamment flexible pour permettre d'y intégrer divers modules d'analyse syntaxique au Paragraphe 2.1. En outre, l'enrichissement des modèles par la syntaxe (Nallapati et al., 2016; Li et al., 2017; Le et al., 2017; Kiperwasser and Ballesteros, 2018) et l'apprentissage par renforcement deviennent des méthodes de choix pour de nombreuses tâches de résumé. Sont-elles redondantes ou bien complémentaires l'une par rapport à l'autre ? Peut-on remplacer l'apprentissage par renforcement par un pré-traitement d'analyse syntaxique ? Empiriquement, quelles connaissances syntaxiques l'apprentissage par renforcement détecte-t-il au cours de son exploration ? Ce sont autant de questions auxquelles nous cherchons à répondre pour améliorer l'explicabilité et l'efficacité des systèmes de résumé automatique neuronaux au Paragraphe 2.2.

2.1 Compression de phrases basées sur l'apprentissage par renforcement

Le modèle séquence-à-séquence (s2s) est un modèle populaire utilisé pour la tâche de synthèse, mais il est encore loin d'être parfait. Tout d'abord, il partage en grande partie les problèmes bien connus de la traduction automatique neuronale (Koehn and Knowles, 2017). En outre, le résumé de texte contient ses propres difficultés auxquelles le modèle s2s doit faire face. Par exemple, Amplayo, Lim, and Hwang, (2018), Cao et al., (2017), Song, Zhao, and Liu, (2018), Li et al., (2018a), and Li et al., (2018b) ont observé que les modèles de séquence peuvent produire une sortie incorrecte, hallucinée et non factuelle. Un remède souvent utilisé consiste à intégrer des biais structurels supplémentaires afin de s'assurer que l'attention du modèle se focalise sur les informations clés de la source. Amplayo, Lim, and Hwang, (2018) ont observé que les informations autour des entités nommées sont liées au thème du résumé. Ils ont proposé d'associer aux entités liées un module de thème pour guider le processus de décodage. Cao et al., (2017) ont utilisé un outil d'extraction d'informations et une technique d'analyse d'arbre de dépendance pour déduire des faits réels à partir du texte source et forcer le générateur à respecter ces descriptions. Dans le même esprit, Song, Zhao, and Liu, (2018) ont exploré l'utilisation des relations syntaxiques issues de l'analyse syntaxique des constituants, Li et al., (2018a) ont utilisé l'algorithme TextRank et Li et al., (2018b) ont exploité des relations d'implication.

En parallèle, une autre voie de recherche vise à apprendre directement à extraire des phrases importantes du document source, ce qui rend le modèle de séquence neurale plus interprétable. Cela s'applique spécifiquement au jeu de données CNN / Daily Mail. Narayan, Cohen, and Lapata, (2018) ont ainsi proposé de définir l'extraction comme un problème de classification et ont utilisé l'apprentissage par

renforcement pour optimiser l'objectif final. Ils soutiennent que la perte d'entropie-croisée n'est pas une métrique appropriée pour cette tâche. Cependant, ce modèle manque d'un composant de réécriture. En s'inspirant de la manière dont l'homme résume un texte, Chen and Bansal, (2018) ont proposé d'unifier les composants d'extraction de phrases et de réécriture au sein d'un unique modèle et ont entraîné un agent pour les optimiser globalement de bout-en-bout.

Nous avons observé que les études précédentes sur les tâches de résumé automatique, en particulier sur le jeu de données Gigaword, utilisent uniquement les analyses en arbre de dépendance comme des biais structurels supplémentaires aux modèles (Cao et al., 2017; Li et al., 2018a; Song, Zhao, and Liu, 2018). Les sous-arbres de la structure de dépendance ne sont toujours pas proprement explorés pour aider à réduire la phrase en entrée en éléments informationnels courts, concis et moins ambigus. Nous voulons étudier différentes façons de sélectionner des sous-arbres pour aider le décodeur à mieux réécrire des fragments de texte. Nous introduisons un modèle (au Chapitre 6) qui exploite l'analyse syntaxique pour extraire des segments cohérents du document source, puis sélectionne le meilleur de ces segments avec un apprentissage par renforcement, et enfin regénère un résumé avec un modèle séquence-à-séquence moderne. Ce modèle peut donc gérer de manière transparente les deux types de résumés extractifs et abstractifs. En outre, notre approche s'éloigne des architectures d'apprentissage profond de bout-en-bout récentes en linéarisant de manière déterministe l'arbre syntaxique en segments de texte superposés à sélectionner avec l'apprentissage par renforcement. Cette approche donne également des propriétés d'adaptabilité et d'explicabilité intéressantes. Adaptabilité, car il est facile de remplacer le module d'analyse syntaxique par un autre composant peu profond, tel que la segmentation, lorsque les analyses d'arbre de dépendance complètes ne sont pas disponibles ou ne sont pas fiables. Explicabilité, car les segments qui mènent aux meilleurs résumés sont clairement identifiés, ce qui n'est pas toujours le cas avec d'autres approches d'apprentissage profond, par exemple basées sur l'attention.

2.2 Résumé enrichi par des connaissances syntaxiques

La modélisation explicite de la syntaxe a souvent été utilisée dans des applications de génération de texte, notamment pour la traduction et le résumé automatique neuronal. Sennrich and Haddow, (2016) enrichissent l'entrée de la traduction automatique neuronale avec des étiquettes de dépendance, des tags des parties du discours, des tags de sous-mots et des lemmes, de sorte que chaque exemple d'entrée soit représenté par l'enchaînement de tous ces plongements. De même, Nallapati et al., (2016) enrichissent l'encodeur d'un modèle de résumé neuronal avec des tags des parties du discours, des tags de reconnaissance d'entités nommées et des caractéristiques TF-IDF. L'intuition est que les mots seront mieux désambiguïsés en tenant compte du contexte syntaxique. En spéculant que des arbres d'analyse complets peuvent être plus avantageux que des informations syntaxiques peu profondes, Li et al., (2017) enrichissent l'entrée avec une linéarisation de l'arbre d'analyse et comparent trois méthodes d'intégration de cet arbre d'analyse (parallèle, hiérarchique et mixte). Le et al., (2017) définissent la traduction automatique en tant que tâche séquence-à-dépendance dans laquelle le décodeur génère les mots et un arbre de dépendance linéarisé, tandis que Kiperwasser and Ballesteros, (2018) proposent une approche d'apprentissage multitâche planifiée dans laquelle la tâche principale est la traduction, mais le modèle est également entraîné sur les étiquettes des parties du discours, des arbres de dépendance et des séquences de traduction.

Simultanément, divers modèles d'apprentissage par renforcement ont été proposés pour les modèles séquence-à-séquence. Ranzato et al., (2016) présentent une adaptation de REINFORCE (Williams, 1992) au modèle de séquence et une stratégie d'apprentissage par curriculum pour alterner les vrais labels et les échantillons du modèle RL. Cette version de base de REINFORCE est connue pour avoir une grande variance. Bahdanau et al., (2016) proposent un autre modèle d'apprentissage par renforcement, dénommé acteur-critique, qui permet d'obtenir une variance inférieure des estimations, compensée par un léger biais. L'impact du biais, ainsi que la qualité du modèle, dépendent en particulier de la conception soignée du *critique*. En pratique, pour assurer la convergence, des techniques complexes doivent être utilisées, notamment un réseau cible supplémentaire Q', un acteur retardé, un terme pénalisant la valeur critique et l'adaptation de la récompense. En revanche, Rennie et al., (2017) présentent un moyen très simple et efficace de construire une meilleure base de référence pour REINFORCE, dénommé la méthode auto-critique d'apprentissage de séquence (SCST). Au lieu de rechercher et de construire une base de référence ou d'utiliser un critique réel comme ci-dessus, SCST utilise sa propre prédiction normalement utilisée au moment de l'inférence pour construire la séquence et l'utilise pour normaliser la récompense.

Les travaux sur le résumé ont largement exploré les modèles de syntaxe et l'apprentissage par renforcement (RL) utilisant ROUGE comme récompense. Cependant, l'impact respectif de ces approches, au-delà de la métrique d'évaluation standard ROUGE qui échoue sans doute à prendre en compte plusieurs aspects qualitatifs importants des textes, n'est pas clair. Malgré tout, le résumé basé sur RL devient de plus en plus populaire. Au Chapitre 7, nous fournissons une comparaison détaillée de ces deux approches et de leur combinaison selon plusieurs dimensions liées à la qualité perçue des résumés générés : combien de mots sont répétés dans le résultat ? La distribution générée des parties du discours est-elle proche de celle de vrais labels ? Quel est l'impact de la durée de la séquence ? Quelle est la qualité de la pertinence sémantique et de la grammaticalité ?

En utilisant la tâche standard de résumé de phrases sur Gigaword, nous comparons SCST à des modèles prenant en compte la syntaxe qui exploitent les parties du discours et/ou les informations d'arbre de dépendance. Nous montrons que sur toutes les évaluations qualitatives, le modèle combiné donne les meilleurs résultats, mais qu'un entraînement avec RL seul, sans aucune information syntaxique, donne des résultats presque aussi bons, avec moins de paramètres et une convergence plus rapide que des modèles avec syntaxe.

3. Conclusion

Bien qu'ils soient très performants, un des inconvénients majeurs des modèles neuronaux est qu'ils sont moins interprétables sur ce qui est appris et sur la manière dont les informations sont codées. Dans les réseaux neuronaux pour le résumé automatique, il est plus difficile de garder trace des informations extraites et sélectionnées que dans les modèles classiques. Dans chaque chapitre de la thèse, nous décrivons comment nous fournissons davantage de preuves empiriques et d'explications à certaines des techniques / méthodes populaires d'analyse du sentiment neuronal et de résumé de texte neuronal. Nous commençons par présenter une revue de l'état de l'art sur le choix de la structure de réseau pour l'analyse de sentiment au Chapitre 2. Ensuite, le Chapitre 3 explore les meilleures topologies possibles de réseaux

de neurones profonds pour cette tâche, en se concentrant sur l'impact de la profondeur du modèle, qui est l'un des éléments clés qui explique le succès de ce type de réseaux en traitement d'image mais est discutable en ce qui concerne le traitement de texte. Dans le Chapitre 4, nous présentons un nouveau modèle neuronal capable de capturer les sentiments dans le contexte d'une conversation en ligne. Le Chapitre 5 décrit plus en détail le travail relatif au résumé de texte, en se concentrant sur deux sujets spécifiques présentant un intérêt particulier dans cette thèse: la compression de phrase et la génération de séquence. Ensuite, nous présentons au Chapitre 6 plusieurs axes de recherche que nous avons explorés pour améliorer à la fois l'efficacité et l'interprétabilité des modèles de compression de phrases neuronales, et nous fournissons une analyse de la qualité des informations syntaxiques capturées par les modèles de résumé les plus courants basés sur l'apprentissage par renforcement au Chapitre 7. Enfin, le Chapitre 8 récapitule et discute quelques difficultés restantes dans l'analyse neuronale des sentiments et dans la recherche en résumé automatique de texte.

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, we are living in the age of intelligent machines. It's hard to ignore because they appear everywhere, either in the form of a physical robots, virtual assistants or autonomous cars. This takes part in advancing the civilization of human kind. With the help of a computer to organize and provide structured information, a child can grow up developping personal skills and acquiring knowledge better. A worker can be more functionnal on his task, less tired in the area that he is less advantageous and has a wider scope of the environnment around him. A society will be harmonious, resources will be distributed more evenly and timely to where they are needed. In addition, people don't just want an *intelligent* machine that can simply respond to a task or give a right answer to a question but also, it should be able to speak, interact and communicate with us readily. Making machines a good supporting friend, humanizing machines for the sake of living and conquest this universe has always been a strong desire that human wants to pursue during the history of evolution.

Indeed, scientists have soon recognized natural language processing (NLP) as a serious subject of studies from the 1950s, though work can be found from earlier periods. In 1950, Alan Turing (Turing, 1950) stated that if a machine can imitate a human so completely that we can not see noticeable differences, then the machine could be considered capable of thinking. This first definition helped inspire, push ideas and evolve the field as a whole. Most notably, an influential proposal by Chomsky, (1957) marked the beginnings of NLP research and the quest of making computer to understand language. In his *Syntactic Structures* book, Noam Chomsky proposed a style of grammar called Phrase-Structure Grammar, which translated natural language sentences into a format that is readable by computers. Since then, NLP has gone through many periods of development and stagnant phases. For several decades, most natural language processing systems used hand-coding rules such as devising heuristic rules by or writing grammars by experts (Winograd, 1971; Schank and Abelson, 1977). This approach showed many disadvantages as it needs a lot of manual work, is time consuming and has less learning capacity. Until the late 1980s, with the steady increase of computational power, there was a revolution in natural language processing with the shift into machine learning algorithms and statistical models. In the 1990s, with the pace of progressing tremendous online text, pure statistics NLP methods N-Grams (Broder et al., 1997) have become useful as a standard way to track clumps of linguistic data numerically. However, language modeling and other learning problems built on this representation are intrinsically difficult to generalize due to the *curse of dimensionality*, i.e. a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training time. Bengio, Ducharme, and Vincent, (2000) introduced a revolutional distributed representation for words learned with a feed-forward neural networks (Rosenblatt, 1958). This brought a lot of moments of excitement and paved the ways

to the development of many NLP applications: Google machine translation (Wu et al., 2016), Amazon Alexa virtual assistant, caption generation (Vinyals et al., 2015),...

More than ever, we are now living in an era where data is growing exponentially. More data brings more opportunities but it also comes with many new problems and great challenges. It is estimated that 90 percent of the human data was generated in just the last two years. When only considering text data, 2 billion active users Facebook update around 734 millions statuses and comments daily. Google now has to process more than 3.5 billion searches per day. Around 23 billion text messages and 225 billion emails are sent to communicate for work and personal aims everyday [1]. This quantity, diversity and velocity of data are unprecedented. Given the increasing pace of incoming data, limited time and human ability of absorbing information, there is an urgent need of creating a powerful tool to extract semantic information from raw texts to assist daily human works. On a general scale, Sentiment Analysis is such a core tool that nowadays, corporations or governments can rarely live without in the information century. In the age of global market exchange with thousands of competitors from diverses sets of cultural differences, Sentiment Analysis is the NLP application that most of the companies will need and think of first. Modern governments need this tool to adapt faster their policy upon the real-time reaction of their citizens on social media. In terms of individuals, the long-goal of building a human-like and friendly machine that can communicate with and assist human in daily life (robots as receptionist, assistants, ...) first needs to master the step of knowing what is the true opinion and sentiment meanings human implying under the word surface. Also, as the time is becoming a scarce factor in the present days, beside extracting sentiment/opinions information, people (and machine) also need to communicate quickly a shorten but faithful summary with the main arguments that support the core ideas in the document.

Recognizing the potential, broad and powerful applications of extracting sentiment and semantic information from text, I'm motivated to work and contribute in the area of Sentiment Analysis and Sentence Compression; specifically, in this thesis, the goal is to advance Neural Sentiment Analysis and provide a better analysis to Neural Text Summarization; new promising approaches developed recently over the past five years with the rise of Deep Learning (Goodfellow, Bengio, and Courville, 2017). Before going into the details of the thesis, we now walk the reader through the background and a bit of the development history of these two tasks.

1.1 Sentiment Analysis

Human behaviours are influenced by opinions coming from the surrounding environment. We construct beliefs and make the choice, to a considerable degree, based on how others see the world. That's why people tends to seek out the opinions of others to make a decision. Since 2000, with the birth of the world wide web and the rapid growing of social networks, when one wants to buy a product, he will not be limited to consult his small number of relatives because there are many user reviews about the product on the internet. This is the same for an organization. It may no longer need to do expensive and long surveys to collect public opinions because abundance of these information are already publicly available. However, looking for such information on numerous sources on the internet with abundant going texts and distilling the opinions in them so that an average human reader

[1] According to Forbes article: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/.

can easily grasp an idea rests a challenging task. Automated Sentiment Analysis, the computational study of people's opinions, sentiments, emotions towards entities and theirs attributes (Liu, 2012), as a result, is an important research area.

Sentiment Analysis mainly studied at three levels: document, sentence and aspect level. Document level sentiment analysis treats the whole document as an information unit while sentence level sentiment analysis classifies each sentence in a document. It is normally formulated as a three-class classification problem: neutral, posivite or negative. Aspect level Sentiment Analysis, on the other hand, is more fine-grained. Given a product review as an example, beside a general opinion of the product, it also extracts positive and negative sentiments on different aspects of the product respectively. Casting as a three-class classification seems to make Sentiment Analysis an easy problem. Notably, with some recent excitements from the rise of Deep Learning, people think that it will be solved "very soon". In fact, Sentiment Analysis still remains an extremely challenging task (Prabha and Umarani Srikanth, 2019; Ain et al., 2017; Liu, 2012; Tang, Qin, and Liu, 2015). Just looking at this phrase for example: *"How can anyone sit through this movie ?"*. Can computers understand and capture the implicit sentiment in the aforementioned phrase without the presence of any sentiment words ? And: *"People should question the stability of mind of the current President of the United States"*. Could computers get the negative sentiment of sarcasm about the President of the U.S ?

1.1.1 Non-Neural Sentiment Analysis

All Sentiment Analysis methods start first with a feature engineering step, which converts a piece of text into a feature vector, in order to enable data driven approaches. Very first methods use binary-valued feature vectors to indicate whether a term occurs (value 1) or not (value 0). Besides, positions of each term are often added to give more sentiment weights to some parts of the text. This proposed feature is due to the observation that words in the titles often reveal more opinions than in the body; first or last few sentences in a text are generally a better indicator of sentiment than elsewhere. To capture a larger context, N-grams (bi-grams and tri-grams) may also be employed in addition to words (unigram) (Dave, Lawrence, and Pennock, 2003). (Hu and Liu, 2004) showed that a sentiment lexicon generated algorithm using a bootstrap strategy can determine the sentiment orientation of a sentence and its aspect-level, given seeds as some positive and negative sentiment words and the list of synonyms and antonyms relations in WordNet. The sentiment of a sentence is then just calculated by averaging up the opinion scores of all sentiment words in the text. In a similar spirit to fight against the data hunger of Sentiment Analysis methods, (Gamon et al., 2005) relied on EM algorithm with Naive Bayes as classifier to learn from a small set of labeled sentences and a large set of unlabeled sentences in a semi-supervised learning approach. (McDonald et al., 2007) learned jointly sentence and document sentiment levels with a hierarchical sequence learning model similar to conditional random fields (Lafferty, McCallum, and Pereira, 2001) to enhance accuracy of both levels of classification.

While these works had a huge impact on the field of SA, reseachers quickly realized that these bag-of-words (BoW) models incur many drawbacks. To better see the ideas, looking at a few lines of the book "A Tale of Two Cities" by Charles Dickens for example:

> *it was the season of Light,*
> *it was the season of Darkness,*
> *it was the spring of hope,*

it was the winter of despair,

If we consider these four lines as an entire document, then we will have a corpus with 24 words and a vocabulary of 11 unique words (ignoring case and punctuation): "it", "was", "the", "season", "of", "light", "darkness", "spring", "hope", "winter", "despair". BoW models transformed each sentence into a numeric feature vector with a fixed length representation equal to the size of the vocabulary. The scoring used can vary from binary to counting the word frequency or even rely on a more balanced scoring TF-IDF methods. As a binary vector, it will look like below.

"it was the season of Light" = [1,1,1,1,1,1,0,0,0,0,0]
"it was the season of Darkness" = [1,1,1,1,1,1,0,1,0,0,0,0]
"it was the spring of hope" = [1,1,1,0,1,0,0,1,1,0,0]
"it was the winter of despair" = [1,1,1,0,1,0,0,0,0,1,1]

Firstly, we see that the vector space is very sparse, each sentence only contains a small number of words in vocabulary when the document gets bigger. Secondly, all ordering of the words are normally discarded, which results in ignoring the semantic meaning of words in the context. Context can contribute a lot to the model, in case if the model needs to differentiate the same words but in another arrangement (*"he is great"* v.s. *"is he great"*). In this aspect, Bag-of-N-grams can help to deal with word order in a short context but it still can not resolve the data sparsity problem.

Regarding the modeling aspect, classical (non-neural) models also show many disadvantages. Let's continue with this BoW representation, namely $\mathbf{x} \in \mathbf{R}^d$ with d as input dimension length, a basic linear classifer of multi-class prediction will be given as:

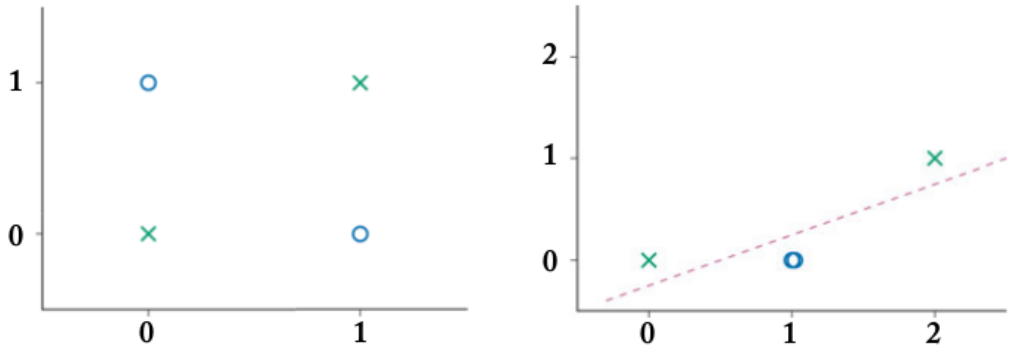
$$\hat{y} = \underset{k}{\text{argmax}} \quad \mathbf{x} \cdot \mathbf{W} + \mathbf{b} \tag{1.1}$$

where the matrix $\mathbf{W} \in \mathbf{R}^{d \times k}$ and vector $\mathbf{b} \in \mathbf{R}^k$ are the parameters of the model. Here, $\hat{y} \in \mathbf{R}^k$ is the score vector assigned by the model to each class $k \in K$. A common treat with this linear classifier is to squash the output of the function of data above $f(\mathbf{x})$ from range $[-\infty, \infty]$ to $[0, 1]$ with logistic function. As such, one can gain more confidence of the decision via obtaining a better interpretation in term of probability assigned to each class:

$$\hat{y} = \underset{k}{\text{argmax}} \frac{e^{(\mathbf{x} \cdot \mathbf{W} + \mathbf{b})}}{\sum_{i=1}^{K} e^{(\mathbf{x} \cdot \mathbf{W} + \mathbf{b})_{[i]}}} \tag{1.2}$$

Linear (or logistic) classifiers only work when the dataset is linearly separable. It is not always the case in practice, considering the illustration of the XOR function in Figure 1.1a for a small example. There are no parameters in 2D that can constitute with this data to draw a straightline separating two classes in blue and green colors. To solve this issue, (Boser, Guyon, and Vapnik, 1992; Shawe-Taylor and Cristianini, 2004) proposed Kernelized Support Vectors Machines (SVMs) methods to transform the original data into another representational space so that we apply a linear separator. In the XOR problem for example, if the feed the data points through the nonlinear function $\phi(x_1, x_2) = [x_1 \times x_2, x_1 + x_2]$, the XOR problem will be solvable (Figure 1.1b). In other words, given the function ϕ, we can train a linear classifier to classify XOR data points:

$$\hat{y} = \phi(\mathbf{x}) \cdot \mathbf{W} + b \tag{1.3}$$



(A) (B)

FIGURE 1.1: (A): XOR problem. (B): XOR problem becomes linearly separable. *Credit: (Goldberg and Hirst, 2017).*

This special pick of function ϕ in the example requires human intuition about the problem and depends particularly on this dataset. In practice, Boser, Guyon, and Vapnik, (1992) and Shawe-Taylor and Cristianini, (2004) generally proposed to map the data into very high dimensional spaces, e.g. via polynomial mapping $\phi(\mathbf{x}) = (\mathbf{x})^d$, to increase the chance of finding an appropriate linear classifier. With $d = 2$ in the XOR problem, the function $\phi(x_1, x_2)$ will consider $(x_1x_1, x_1x_2, x_2x_1, x_2x_2)$. These all combinations of two variables come with an increase of polynomial speed of the number parameters to solve. This is computationally infeasible for most of the NLP applications based on rich word-embedding features. (Schölkopf et al., 2001) proposed to use *kernel trick* for kernel methods but it still scales linearly with the size of the training set, preventing the model from training on large dataset. Another approach is to let the training algorithm find and train this appropriate representation in addition to the linear classifier. For example, a parameterized linear model followed by a nonlinear activation g can take place of the mapping function:

$$\phi(\mathbf{x}) = g(\mathbf{x} \cdot \mathbf{W}' + \mathbf{b}') \tag{1.4}$$

The entire prediction $g(\mathbf{x} \cdot \mathbf{W}' + \mathbf{b}')\mathbf{W} + \mathbf{b}$ is non-convex, differentiable and can be trained via gradient-based techniques to learn both the mapping function and the linear classifier. This is the motivating idea of neural models described more in detail in the next section.

1.1.2 Neural Sentiment Analysis

Neural Sentiment Analysis is a new approach that addresses the aforementioned problems. First, Neural Sentiment Analysis consists of a neural network architecture with multiple layers which allows to exploit well the representation of the data when the network gets bigger and bigger. A small example of a network with one data layer and a hidden layer is shown in the Figure 1.2 below. To tackle the high dimensionality of BoW, Neural Sentiment Analysis relies on word embedding (Bengio, Ducharme, and Vincent, 2000; Collobert et al., 2011), which is low dimensional dense vectors to encode semantic and syntactic properties of words. In the word embedding space, feature vectors of similar words end up in the same location; syntactic transformation of *"king"* and *"queen"* to *"man"* and *"woman"* can be expressed nearly equivalently through linear relationships between their vectors (Mikolov et al., 2013). With these fascinating properties, it is believed that Neural Sentiment

Analysis in particular, or Neural Network methods (or Deep Learning methods) in general, can generalize and leverage better with more data, which results in a surpassing of traditional machine learning methods in Figure 1.3.

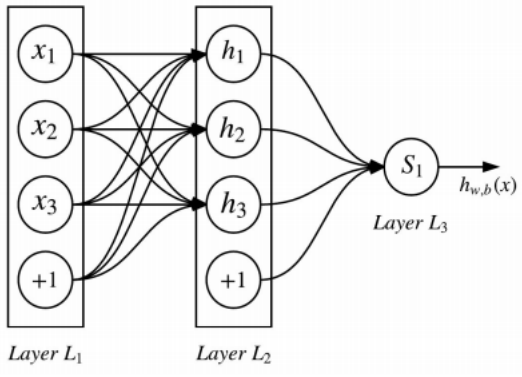


FIGURE 1.2: Feedforward neural network.

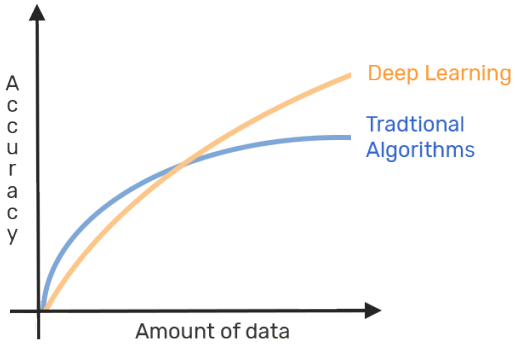


FIGURE 1.3: Performance of Deep Learning vs. traditional machine learning methods.

While word embeddings like CBOW or Skip-gram (Mikolov et al., 2013) capture better the context than BoW, words with similar contexts can still express opposite sentiment polarities. As a results, they may be projected to a nearby region in the embedding space. Therefore, (Maas et al., 2011; Tang et al., 2014b) proposed variations of word embedding that can get both sentiment and semantic information. (Bespalov et al., 2011) showed that latent representation combined with N-grams can generate a more appropriate embedding for sentiment classification. Besides, multi-sense word embeddings are also explored. (Vo and Zhang, 2015) investigated aspect-based sentiment classification with automatic features coming from unsupervised learning techniques; (Ren et al., 2016) introduced methods to learn topic-enriched multi-prototype word embeddings on Twitter data.

Neural network architecture is also an important topic of research in Neural Sentiment Analysis. In the early days of using neural networks models, researchers tend to believe that explicit syntactic and semantic information like parse trees, opinion lexions, part-of-speech tags... can help to infer better the sentiment composition of words in a short text. (Socher et al., 2011) first introduced a semi-supervised Recursive Autoencoders Network (RAE) to get a vector representation of a sentence for sentence sentiment classification. Later then, (Socher et al., 2012) proposed a Recursive Neural Tensor Network (RNTN) in order to predict more accurately the compositional semantic effects of words with the help of a fully labeled parse tree

of sentences obtained from an external parser (in Figure 1.4). (Qian et al., 2015) realized that a composition of tag embeddings would make word embeddings more effective, they proposed thus two models Tag-guided Recursive Neural Network (TG-RNN) and Tagembedded Recursive Neural Network (TE-RNN).

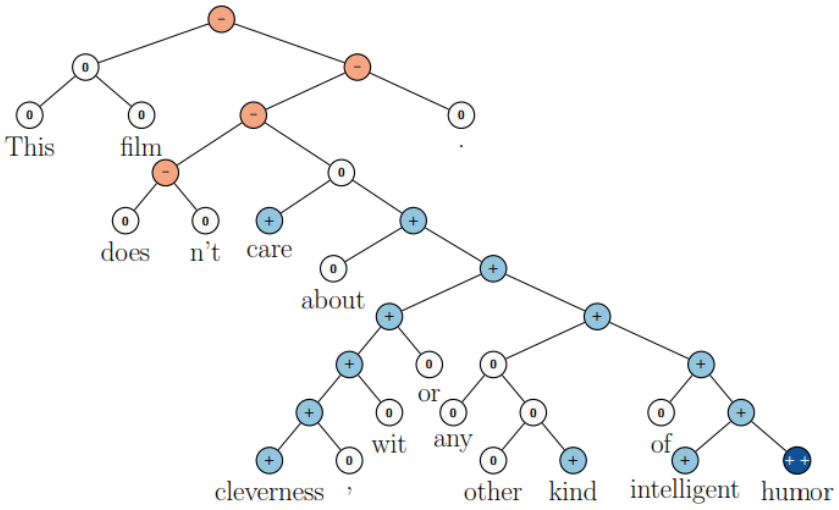


FIGURE 1.4: Example of the Recursive Neural Tensor Network accurately predicting 5 sentiment classes, very negative to very positive (– –, –, 0, +, + +). *Credit: (Socher et al., 2013).*

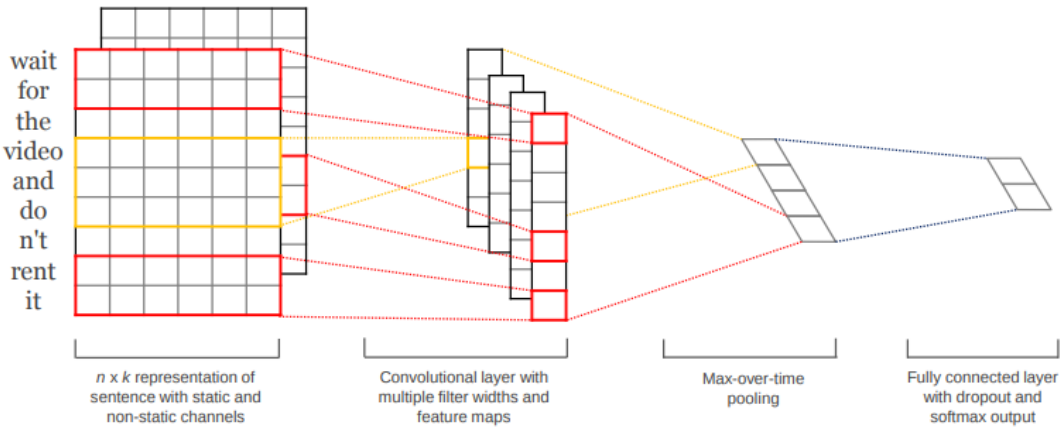


FIGURE 1.5: A Convolutional Neural Network for sentiment classification. *Credit: (Kim, 2014).*

These approaches first showed a lot of potential benefits of neural networks for Sentiment Analysis but with a disadvantage that they always need to parse trees to extract features from sentences. With a particularly careful design of models, a simpler and yet, effective Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) model is later found to be able to learn intrinsic relationships between words too. These models are appealing as they can be trained end-to-end directly from the learning objective; hence, eliminating unnecessary components and enabling better leveraging of input data. (Kalchbrenner, Grefenstette, and Blunsom, 2014) introduced a Dynamic CNN (DCNN) which uses a dynamic k-max pooling operator as a non-linear subsampling function for semantic modelling of sentences. (Kim, 2014) experimented multiple variants of CNN-rand, CNN-static, CNN-non-static for sentence sentiment classification (in Figure 1.5). (Wang, Jiang, and Luo,

2016) proposed a joint CNN and RNN architecture to take advantage of the long-distance dependencies learned via RNN and the coarse-grained local features captured by CNN.

Despites these advances, researches on Sentiment Analysis show divergences about the necessity of applying a deep network structure for sentiment analysis and text classification tasks (Joulin et al., 2016; Kim, 2014; Zhang, Zhao, and LeCun, 2015; Conneau et al., 2016). If the utility of taking the deep paradigm has been clearly confirmed in computer vision domain, we aim to provide a similar empirical conclusion for the NLP domain. Besides, the majority of works on neural sentiment classifiers still rely on raw text features, which are clearly not sufficient to modelize human opinions and behaviors. An example of useful context is dialog acts in the context of the conversation, which are not often explored thoroughly for sentiment analysis tasks. We want to contribute to build a stronger neural sentiment model which can be more explainable and adapt better to a more complex natural language processing task such as conversation modeling.

1.2 Automatic summarization

Extracting sentiment information from digital documents on the Internet enables readers to quickly capture some of the relevant information they are interested in these texts. However, even when done at the aspect and topic level, there is still a lot of additional useful semantic information that is contained in the texts and that is not readily available to the readers. For practical concrete applications, the Sentiment Analysis task is thus rarely sufficient alone, and other texts analysis approaches must be considered to help people understand the important content without having to read in details the whole documents themselves. Text Summarization, the reduction of a text to its essential content, is thus an interesting task that can supplement Sentiment Analysis in answering the questions *"What"*, *"Why"* and *"How"*. With the current non ceasing speed of accumulating data, this could also reduce human reading time, help to accelerate the searching for a document and facilitate the communication with users via automatic assistant such as Google Home, Amazon Alexa,... A quick overview of the human process of summarizing can be depicted in Figure 1.6. To achieve this task, a certain level of linguistic competence, world knowledge and intelligence is required even for humans to perform well: multiple fragments of a text must be chosen, rewritten and gathered according to their relevance while taking into account the coherence, cohesion as well as the temporality of the information included in the summary. Thus, this is a difficult cognitive task. Starting with the pionnering work of (Luhn, 1958) in the 1950s, the field of Text Summarization has evolved but the question of how we can determine the relevance of information included in documents and measure the representativeness and the significance of the information still remain major challenges for automatic summarization.

As many different input types exist, Text Summarization is categorized into single document, multidocument (summary of a group of documents about a specific topic) or domain-specific summarization respectively. Depending on the purpose, Text Summarization can be processed to generate generic or query-guided summary. Generic summarization summarizes documents regardless of its topic or domain while Query-based summary focuses on only what are queried from the source text. An example of query-based application are snippets produced by search engines

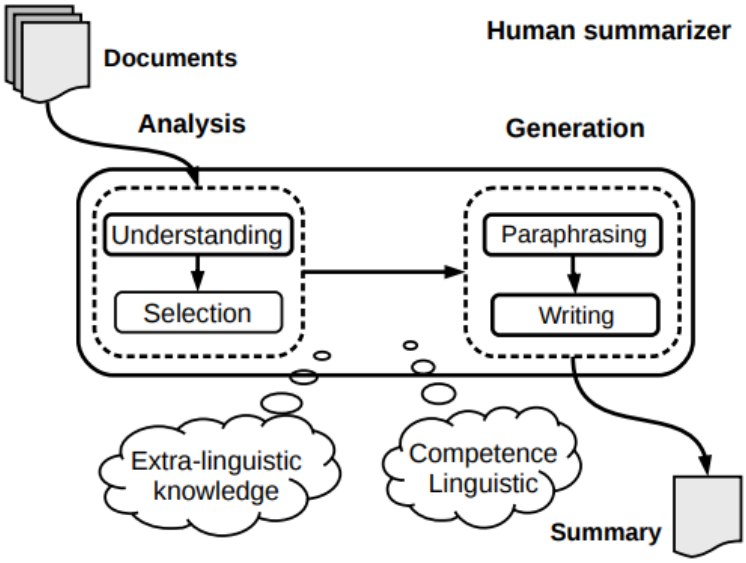


FIGURE 1.6: Human summarization. *Credit: (Moreno, 2014).*

(Nenkova and McKeown, 2011). Depending on the applicative target of the output type, Text Summarization is classified into extractive and abstractive methods. Extraction is the assembly of fragments which contain essential information from a source document. Though unlike the human process, extraction is easier to implement and can still be served as an overview of the original text's content. Abstraction seeks to produce summaries, which are not just a copy of some sentences from the input text, by reformulating and fusing ideas and often by expressing them in other words. Abstracted texts look more natural and better mimic the human behavior but are also very challenging to produce. Abstractive summarization is thus the ultimate goal of text summarization research at the current time. A typical machine summarizer will look like in Figure 1.7.

1.2.1 Non-Neural Text Summarization

As described earlier, extractive summarization techniques choose a subset of the sentences in the source text to produce summaries. More specifically, all extractive summarizer always follow these three steps (Nenkova and McKeown, 2011):

1. Build an intermediate representation of the main information of the input text.
2. Once the intermediate representation is generated, each sentence is assigned an importance score based on its representation.
3. Eventually, to produce a summary, a summarizer system uses greedy algorithms or select the top k most important sentences.

Some other approaches can transform the selection process into an optimization problem where a subset of sentences is chosen while maximizing the overall importance and coherency and minimizing the redundancy. Algorithms to build intermediate representations for extraction are prolifics. Generally, they can be classified into two major types of approaches based on the representation: topic representation and indicator representation. Topic representation approaches transform the text into topic(s) to falicitate their interpretation in the text. Techniques employed are

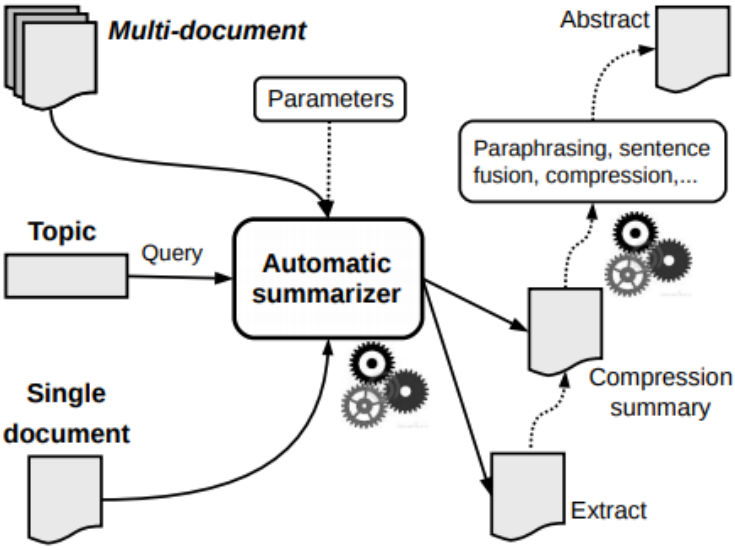


FIGURE 1.7: Machine summarization. *Credit: (Moreno, 2014).*

mostly: topic word approaches, latent semantic analysis and Bayesian topic models (Nenkova and McKeown, 2011). On the other hand, indicator representation approaches describe each sentence by a set of features (indicators) of importance (i.e.: length of sentence, position in the document, having certain phrases,...) and use them to rank directly sentences. Models used could be Naive Bayes, decision trees, support vector machines, Hidden Markov models and Conditional Random Fields. For more information on extractive techniques, we refer the audience to an excellent review by (Allahyari et al., 2017).

Source: The Los Angeles Dodgers acquired South Korean right-hander Jae Seo from the New York Mets on Wednesday in a four-player swap.
Summary: Korea's Seo **headed to** Dodgers from Mets.

TABLE 1.1: An example on abstractive summarization taken verbatim from (Amplayo, Lim, and Hwang, 2018). The boldfaced words in the summary do not appear in the input document.

Compared to extractive summarization, abstractive summaries are arguably a better approximation of human summaries (Cohn and Lapata, 2008). Table 1.1 shows an example of a human-generated abstractive summary. Early techniques for abstractive summarization include sentence fusion (Filippova and Strube, 2008), which used local multi-sequence alignment to identify expressions containing the same information, and statistical generation to group common phrases into a sentence. Later, (Cohn and Lapata, 2009) proposed Tree Transduction method for Sentence Compression, which allows to construct a grammatical summary of a given sentence. (Tanaka et al., 2009) introduced a sentence revision method, which synthesizes information across sentences via insertion and substitution of the leading phrases.

The aforementioned approaches show little improvement over extractive methods, however. This leads to the development of a fully abstractive pipeline, which often consists of three subtasks: information extraction, content selection, and surface realization. First, important information is extracted from the input text such as noun phrases (NPs) and verb phrases (VPs) with their contextual information

(Bing et al., 2015). Then, like extractive techniques, a subset of candidate phrases is selected by heuristic methods or by Integer Linear Programming (ILP) (Woodsend and Lapata, 2011), which offers a joint optimization (and not independent) of selection of phrases with regard to a set of linear constraints. Finally, these selected candidates are combined to generate a summary using grammatical/syntactic rules. An example of such surface realization techniques is SimpleNLG (Gatt and Reiter, 2009). Other methods like Graph-based and Template-based are presented in (Lin and Ng, 2019).

Although generating a summary is difficult, evaluating its quality is another challenging task itself. Unlike Sentiment Analysis, in Text Summarization, the output of the system is natural language text. How can we know if a produced summary is correct ? How can we decide whether a summary is of higher quality than another ? Which respective ratios of the content, length and quality of a summary can be considered as acceptable ? Moreover, cohesion, which is the linguistic level of the summary, and coherence, its semantic level with regard to the source document must also be respected. Since the early 2000s, ROUGE (Lin, 2004) is introduced as an attempt to judge the quality of the summary and has remained the most widely used metric for automatic evaluation ever since. Essentially, ROUGE works by comparing the number of n-grams in a candidate summary against a set of reference summaries. As depending on the comparison of n-grams, ROUGE suffers from the bias of large co-occurrences number of unmeaningful stopwords (articles, conjunctions, pronouns, etc.). Further, this word-overlap metric fails to capture the semantic similarity between the model and the reference when there are few or no common words. PYRAMID (Nenkova and Passonneau, 2004) and Meteor (Banerjee and Lavie, 2005) are introduced to resolve parts of these deficiencies by proposing more advanced matching strategies based on stemmed forms and synonyms and a weighted semantic matching of content units. Automatic evaluation is still an open problem to which reseachers have sought to respond with partial answers until now.

1.2.2 Neural Text Summarization

In recent years, due to significant advances in the development of neural methods, focus of Text Summarization has gradually shifted from extractive techniques to abstractive techniques. In classical abstractive methods, each component needs to be tuned separately, e.g., information extraction, content selection, and surface realization, which makes it difficult to combine them together and to innovate. As a result, the summarization quality has stagnated and major changes to the existing frameworks were indeed need. Neural Text Summarization has been a new approach that addresses the aforementioned problems. Neural Text Summarization requires minimal domain knowledge, just a parallel corpus of source and summary pairs and it can be trained end-to-end directly from the learning objective without having to learn multiple different components. Rush, Chopra, and Weston, (2015)'s work of applying Neural Machine Translation to abstractive summarization is the first to spark a novel way of building abstractive summarizers. Since then, this neural sequence-to-sequence (seq2seq) (Figure 1.8) (Sutskever, Vinyals, and Le, 2014) based model has become the core technology for most of the modern abstractive systems.

Neural Text Summarization summarizes as follows: an encoder reads through the given source sentence and encodes as a list of fixed-length vector representations that represents the sentence meaning; a decoder then processes the encoded

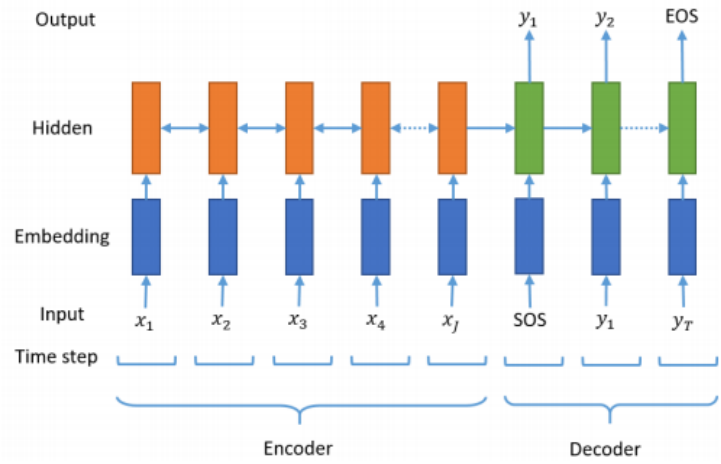


FIGURE 1.8: A basic sequence to sequence model

vectors to produce a summary. Encoding plays a similar role as information extraction in classical approaches. Nevertheless, encoding long documents with Recurrent Neural Network (RNN) (like in Figure 1.8) or even Convoluational Neural Network (CNN) without losing essential information is difficult with this approach. Pre-selecting representative sentences via leveraging extractive methods to distill a long input text into a more compact representation (Chen and Bansal, 2018) is used as one way to tackle this problem. Unlike the encoder, a decoder is originally and commonly implemented using an RNN. At each timestep, the RNN takes two vectors as input, a vector of the previously generated words and a hidden representation vector of the input sequence, to produce an output vector with the size of the vocabulary, which is then converted by a softmax layer into a distribution of vocabulary words. Under this distribution, the most probable word is produced as the output or more effectively, the k-best paths (where k is the beam size) are determined via a beam search (Rush, Chopra, and Weston, 2015).

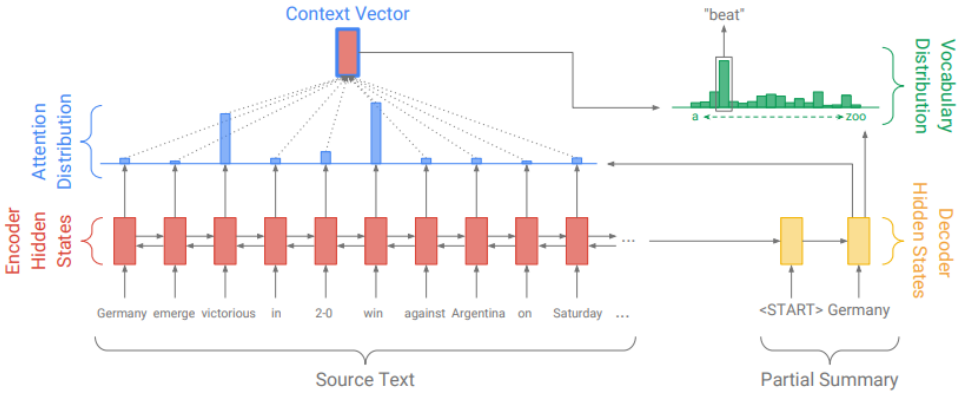


FIGURE 1.9: A seq2seq model with attention and pointer. *Credit: (See, Liu, and Manning, 2017).*

Numerous attempts have been made to improve the seq2seq framework for abstractive summarization. Noticing that some important phrases are more likely to appear than others in a summary, Bahdanau, Cho, and Bengio, (2014) proposed an attention vector to identify important phrases. This attention vector, which is fed to the decoder with an extra context vector from the input sequence, can emphasize and de-emphasize the selection of certain information in the input (corresponding

to high and low masses in the attention distribution). Though subsequently commonly employed, this attention mechanism tends to pick frequent words in practice. Moreover, neural seq2seq is well-known to be weak for generating rare and out-of-vocabulary (OOV) words. To remedy this issue, Vinyals, Fortunato, and Jaitly, (2015) introduced a pointer network, which can copy phrases directly from the input to the output and allows the model to focus on those important rare or OOV words. Figure 1.9 shows an enhanced version of seq2seq with these two popular components: attention and pointer. Nowadays, Neural Text Summarization evolves in the direction of mixing extractive and abstractive techniques for producing a summary, which arguably mimics better our human behavior.

Though being popular, sequence-to-sequence with attention models can not handle very well long sequences in practice. Moreover, the sequential structure of RNN makes it hard to get the full benefit of GPU parallelization. Recently, Vaswani et al., (2017) proposed Transformer with multi-headed self-attention mechanism that allows layer outputs processed in parallel and distant words can affect other words' representation without going through many recurrent steps. Thus, this model can learn and perform very fast and efficiently on large text document, which was still a challenge for many seq2seq based models. This transformer architecture is receiving a lot of interest from the community as it is the base of influential language models like BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019).

Sequence-to-sequence based text summarization model revives the hope of constructing human-like abstractive summary. However, that wish is still far from our reach as it still encounters numerious problems: performing unsatisfactorily on long sequences (Chen and Bansal, 2018), often generating hallucination and unfaithful facts (Amplayo, Lim, and Hwang, 2018; Cao et al., 2017; Song, Zhao, and Liu, 2018; Li et al., 2018a; Li et al., 2018b), ... to name a few. Besides, enriching model with syntax (Nallapati et al., 2016; Li et al., 2017; Le et al., 2017; Kiperwasser and Ballesteros, 2018) and reinforcement learning training, rewards are becoming ubiquitous on numerous summarization tasks. Are those efforts duplicated or complementary to each other with regard to the final syntactic aware result ? Can reinforcement learning replaces the need of heavily preprocessing parsing features ? Empirically, which syntactic structures reinforcement learning is detecting during exploration ? These are all questions that we seek to answer to improve neural summarizers.

1.3 Thesis Outline

Despite of being high performing, a major drawback of neural models is that they are arguably less interpretable about what is learned and how information is encoded. In neural summarization networks, it is more difficult to keep track of the information that has been extracted and selected than in classical models. In each chapter of this thesis, we will describe how we provide more empirical evidences and explanations to some of the popular techniques/methods of neural sentiment analysis and neural text summarization. In brief, this thesis is organized as follows. We start off by providing a review of the literature on the choice of the network structure for Sentiment Analysis in Chapter 2. Then, Chapter 3 explores the best possible topologies of deep neural networks for this task, focusing on the impact of the depth of the model, which is one of the key element that explains the success of this type of networks in image processing, but is questionable when text processing is concerned. In Chapter 4, we present a novel neural model that can capture

sentiment in the context of an online conversation. Chapter 5 describes more in detail the related work on text summarization, focusing on two specific topics that are of particular interest in this work: sentence compression and sequence generation. Then, we present in Chapter 6 several research directions that we have explored to improve both the effectiveness and interpretability of neural sentence compression models, and we provide quality analysis of syntactic implications of most popular reinforcement learning based summarization models in Chapter 7. Chapter 8 wraps up and discusses some remaining challenges in neural sentiment analysis and text summarization research.

The general outline of the thesis is thus structured into two parts: one about sentiment analysis and another about text summarization, each one composed of two main contributions that are briefly summarized as follows:

Part 1, Chapter 3: Networks depth for sentiment analysis

Neural network models are good at learning hierarchical features representation from natural data. Similar to computer vision domain, several works of (Zhang, Zhao, and LeCun, 2015; Schwenk et al., 2017) have attempted to build a very deep convolutional neural network for text classification task. Nevertheless, the results obtained on two distinct data types with different natures of the representation are not conclusive and do not lead to a clear answer to this question. In Chapter 3, we propose a new adaptation of the deepest CNN called DenseNet for text classification and study the importance of depth in convolutional models with different atom-levels (word or character) of input. Our experiments on five standard text classification and sentiment analysis tasks show that deep models indeed give better performances than shallow networks when the text input is represented as a sequence of characters. However, a simple shallow-and-wide network outperforms the deep models DenseNet with word inputs. Our shallow word model further establishes new state-of-the-art performances on two datasets: Yelp Binary (95.9%) and Yelp Full (64.9%).

Part 1, Chapter 4: Sentiment analysis in dialogues

One of the most studied natural language processing task on Twitter is sentiment recognition. Surprisingly, there are fewer works about sentiment analysis that study dialogues on Twitter and a majority of works ignores the relation between them. Going from the speculation that dialog acts can be a factor of explanation and are correlated to sentiment analysis on social media, we build in Chapter 4 a new micro-blog corpus from Mastodon, a relatively large refederated Twitter-like social medium with open interfaces. Specifically, we have manually annotated both dialogues and sentiments on this corpus, and train a multi-task hierarchical recurrent network on joint sentiment and dialog act recognition. We experimentally demonstrate that transfer learning may be efficiently achieved between both tasks, and further analyze some specific correlations between sentiments and dialogues on social media. Both the annotated corpus and deep network are released freely to the community with an open-source license.

Part 2, Chapter 6: Reinforcement learning-based sentence compression

Sentence compression involves selecting key information present in the input and rewriting this information into a short, coherent text. While dependency parses have often been used for this purpose, we propose in Chapter 6 to exploit such syntactic information within a modern reinforcement learning-based extraction model. Furthermore, compared to other approaches that include syntactic features into deep learning models, we design a model that has better explainability properties and is flexible enough to support various shallow syntactic parsing modules. More specifically, we linearize the syntactic tree into the form of overlapping text segments, which are then selected with reinforcement learning and regenerated into a compressed form. Hence, despite relying on extractive components, my model is also able to handle abstractive summarization. We explore different ways of selecting subtrees from the dependency structure of the input sentence and compare the results of various models on the Gigaword corpus.

Part 2, Chapter 7: Enriching summarization with syntactic knowledge

Work on summarization has explored both reinforcement learning (RL) optimization using ROUGE as a reward and syntax-aware models, such as models whose input is enriched with part-of-speech (POS)-tags and/or dependency information. However, it is not clear what is the respective impact of these approaches beyond the standard ROUGE evaluation metric, which arguably fails to capture several important qualitative aspects of texts. Nevertheless, RL-based for summarization is becoming more and more popular. Chapter 7 provides a detailed comparison of these two approaches and of their combination along several dimensions that relate to the perceived quality of the generated summaries. Using the standard Gigaword sentence summarization task, we compare an RL self-critical sequence training (SCST) method with syntax-aware models that leverage POS tags and/or Dependency information. We show that on all qualitative evaluations, the combined model gives the best results, but also that only training with RL and without any syntactic information already gives nearly as good results as syntax-aware models with less parameters and faster training convergence.

1.3.1 Thesis contributions

The main contributions of this work that have been accepted for publication can be summarized into the following list:

- We proposed a new adaptation of the deepest CNN called DenseNet (Huang et al., 2016) for text classification and studied the importance of depth in convolutional models with different atom-levels (word or character) of input. We provided thorough empirical results which showed that deep convolutional networks are not really necessary for word-level text classification, unlike in the computer vision domain. *This work is based on the paper (Le, Cerisara, and Denis, 2018), in which I am the main contributor.*

- We presented a multi-task hierarchical recurrent network to jointly model sentiment and dialog act recognition, a topic which is currently not much explored in sentiment analysis. We showed that transfer learning may be efficiently achieved between both tasks, and further analyzed some specific correlations

between them. We also annotated and proposed a new Twitter-like microblog corpus from Mastodon, which is made available to the community with an open-source license. *This work is based on the paper (Cerisara et al., 2018), in which I have contributed in the model and corpus design phase and co-supervision of students*.

- We proposed a new way to exploit syntactic information during compression process within a modern reinforcement learning-based extraction-abstraction model (Chen and Bansal, 2018). We introduced and explored multiple methods to obtain segments of sequence input via linearizing its syntactic tree to allow better explainability and flexibility of the popular sequence-to-sequence framework. *This work is based on the paper (Le, Cerisara, and Gardent, 2019), in which I am the main contributor.*

- We provided a detailed comparison of two popular approaches of syntax and RL-based and their combination along several dimensions that relate to the perceived quality of the generated summaries. Using the standard Gigaword sentence summarization task, we showed that only training with RL and without any syntactic information already gives nearly as good results as syntax-aware models with less parameters and faster training convergence. *This work is based on the paper (Le, Cerisara, and Gardent, 2019), in which I am the main contributor*.

Part I. Sentiment Recognition

Chapter 2

Related Work

2.1 Neural Networks architectures for sentiment analysis

Text classification is an important task in Natural Language Processing. Traditionally, linear classifiers are often used for text classification (Joachims, 1998; McCallum and Nigam, 1998; Fan et al., 2008). In particular, (Joulin et al., 2016) show that linear models may scale to a very large dataset rapidly with a proper rank constraint and a fast loss approximation. However, a recent trend in the domain is to exploit deep learning methods, such as convolutional neural networks: (Kim, 2014; Zhang, Zhao, and LeCun, 2015; Schwenk et al., 2017) and recurrent networks: (Yogatama et al., 2017; Xiao and Cho, 2016). Sentiment Analysis is also an active topic of research in NLP for a long time, with real-world applications in market research (Qureshi, O'Riordan, and Pasi, 2013), finance (Bollen, Mao, and Zeng, 2011), social science (Dodds et al., 2011), politics (Kaya, Fidan, and Toroslu, 2013). The SemEval challenge has been setup in 2013 to boost this field and is still bringing together many competitors who have been using an increasing proportion of deep learning models over the years (Nakov et al., 2013; Rosenthal et al., 2014; Nakov et al., 2016a). 2017 is the fifth edition of the competition, with at least 20 teams (over 48 teams) using deep learning and neural network methods. The top 5 winning teams all use deep learning or deep learning ensembles. Other teams use classifiers such as Naive Bayes classifier, Random Forest, Logistic Regression, Maximum Entropy and Conditional Random Fields (Rosenthal, Farra, and Nakov, 2017).

Convolutional neural networks with end-to-end training have been used in NLP for the first time in (Collobert and Weston, 2008; Collobert et al., 2011). Their architecture is summarized in Figure 2.1. In a nutshell, they successively take a sentence and pass it to a lookup table layer, convolutional layers produce local features around each word, then these features are combined into a global feature vector which is then passed to final standard layers. The features through many stacked layers are automatically learned and trained by backpropagation following a predefined task. Formally, every word $w \in Vocab$ is encoded as a d-dimensional feature vector using a lookup table layer $Lookup_W(.)$:

$$Lookup_W(w) = \mathbf{W}_w, \tag{2.1}$$

where $\mathbf{W} \in \mathbf{R}^{d \times |Vocab|}$ is the embedding matrix to be learned, $\mathbf{W}_w \in \mathbf{R}^d$ is the w^{th} column of $\mathbf{W}$ and d is the number of embedding space dimensions (a hyper-parameter determined by the user). The first layer of the model transforms each word $[w]_i$ in an input sentence $\{w_1, w_2, ..., w_N\}$ of length N (namely $[w]_1^N$) into a series of vectors, producing the following output matrix:

$$Lookup_W([w]_1^N) = \left(\mathbf{W}_{[w]_1}\mathbf{W}_{[w]_2}...\mathbf{W}_{[w]_N}\right). \tag{2.2}$$

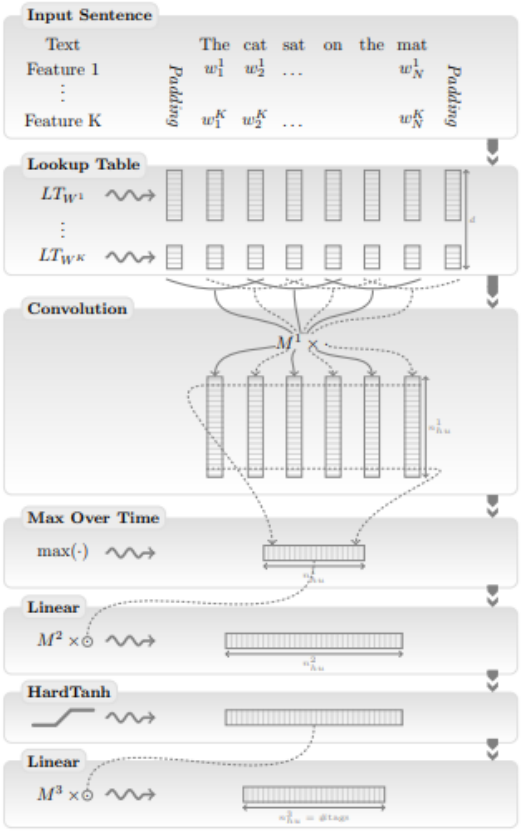


FIGURE 2.1: A Convolutional neural network for sentence modelling.
Credit: (Collobert et al., 2011).

Further, due to this facilitation of automatic learning representation, one can add features other than words if they are helpful for the task of consideration. For example, for the text classification task, one could use the Part-of-Speech tags to disambiguate the meaning of words in different contexts. One can even use features such as lemmas, dependency parsing tags, upper/lower case,... to make model aware of these aspects. Generally speaking, a word can be represented by K discrete features $w \in Vocab^1 \times ... \times Vocab^K$, where $Vocab^k$ is the dictionary for the k^{th} feature. Each feature is associated with a lookup table $Lookup_{W^k}(.)$, with parameters $\mathbf{W}^k \in \mathbf{R}^{d^k \times |Vocab^k|}$. For a sequence of words $[w]_1^N$, the matrix output of the lookup table layer are obtained by concatenating all lookup table outputs. This is similar to Equation 2.2 with extra rows are appended for each discrete feature:

$$Lookup_{W^1,...,W^K}([w]_1^N) = \begin{pmatrix} \mathbf{W}^1[w_1]_1 & \dots & \mathbf{W}^1[w_1]_N \\ \vdots & & \vdots \\ \mathbf{W}^K[w_K]_1 & \dots & \mathbf{W}^K[w_K]_N \end{pmatrix}. \tag{2.3}$$

This matrix, which is typically randomly initialized, can then be fed into further convolutional network layers. A convolution filter $\mathbf{w}_c \in \mathbf{R}^{h \times d}$ (with matrix-vector operation) will be applied to each successive h-grams $w_{i:i+h-1}$ in sequence.

$$c_i = f(\mathbf{w}_c \cdot \mathbf{w}_{i:i+h-1} + b_c) \tag{2.4}$$

with $b_c \in \mathbf{R}$ a bias term and f a non-linear function, for instance the ReLU. This

produces a *feature map* $\mathbf{c} \in R^{N-h+1}$, where h is the size of the window. Convolutional layers are often stacked to produce higher level features, where each layer is followed by a non-linear activation function (Figure 2.2). We can observe that one convolutional layer with window size $h = 2$ results in aggregating bi-grams representation: *"the actual"*, *"actual service"*,... The following convolutional layer covers a broader and more complex scope of three uni-grams (i.e. *"the"*, *"actual"*, *"service"*) and two bi-grams (i.e. *"the actual"*, *"actual service"*) thanks to the hierarchical stacking structure. One can also define how much we want to shift the filter at each step. In Figure 2.2 the stride side was 1, and consecutive applications of the filter overlapped. A larger stride size can lead to fewer filter application and a smaller output size. Figure 2.3 shows stride sizes of 1, 2 and 3 applied to one-dimensional input.

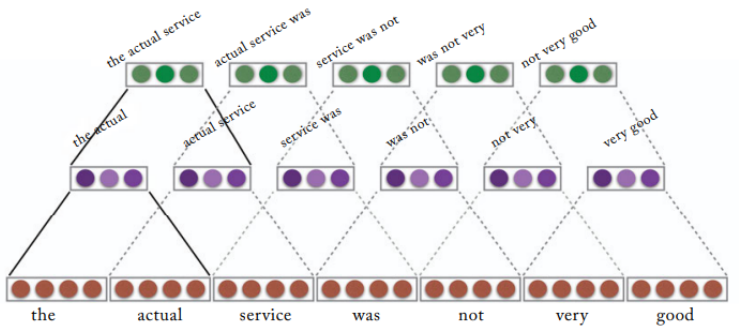


FIGURE 2.2: Two-layer hierarchical convolution with window size $h = 2$. *Credit image: (Goldberg and Hirst, 2017).*

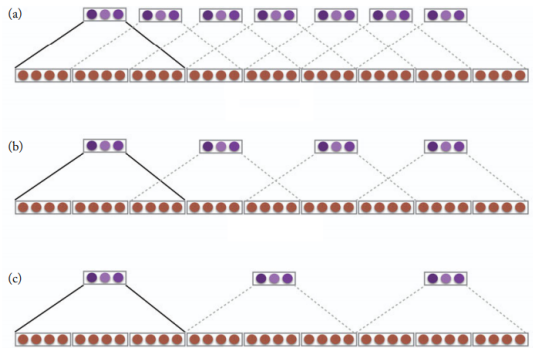


FIGURE 2.3: (a–c) Convolution layer with window size $h = 3$ and different stride sizes 1, 2, 3. *Credit image: (Goldberg and Hirst, 2017).*

These local feature vectors produced by the convolutional layers are then combined in order to obtain a global feature vector through a dimensionality reducting operation, called Pooling, which enables the subsequent application of fixed-size fully connected layers (LeCun et al., 1998). Pooling operations may compute either the average or maximum value per dimension over "time" of the sequence. Collobert et al., (2011) used a max approach, which guides the network to pick the most interesting local features given by the convolutional layers:

$$\hat{c} = \max\{\mathbf{c}\} \in \mathbf{R} \tag{2.5}$$

This process is repeated to obtain m filters with different window sizes h.

$$\mathbf{g} = [\hat{c}_1, \hat{c}_2, \cdots, \hat{c}_m] \tag{2.6}$$

Finally, a standard affine layer is applied:

$$z = f(\mathbf{w}_y \cdot \mathbf{g} + b_y) \tag{2.7}$$

This vector z is then passed to a classification layer that applies a *softmax* activation function (Bridle, 1990) to compute the predictive probabilities for all Q target labels:

$$p(y = q|Z) = \frac{\exp(\mathbf{w}_q^T \mathbf{z} + b_q)}{\sum_{q'=1}^{Q} \exp(\mathbf{w}_{q'}^T \mathbf{z} + b_{q'})} \tag{2.8}$$

where the weight and bias parameters $\mathbf{w}_q$ and b_q are trained simultaneously with the main model's parameters. The loss function to minimize is the cross-entropy error.

Inspired by this seminal work, Kim, (2014) proposed a simpler architecture with slight modifications of Collobert and Weston, (2008) consisting of fine-tuned or fixed pretraining word2vec embeddings (Mikolov et al., 2013) and their combination as multi-channel. Figure 2.4 shows the architecture of the model where a global max-pooling is applied to 3 convolutional layers with kernel window sizes 3,4,5 and then the outputs of each kernel are concatenated to a unique vector to feed to a fully connected layer. The author showed that this simple shallow-and-wide model can already achieve state-of-the-art performances on many small datasets comparing to a deep network. Kalchbrenner, Grefenstette, and Blunsom, (2014) later proposed a dynamic k-max pooling to handle variable-length input sentences. This dynamic k-max pooling is a generalisation of the max pooling operator where k can be dynamically set as a part of the network.

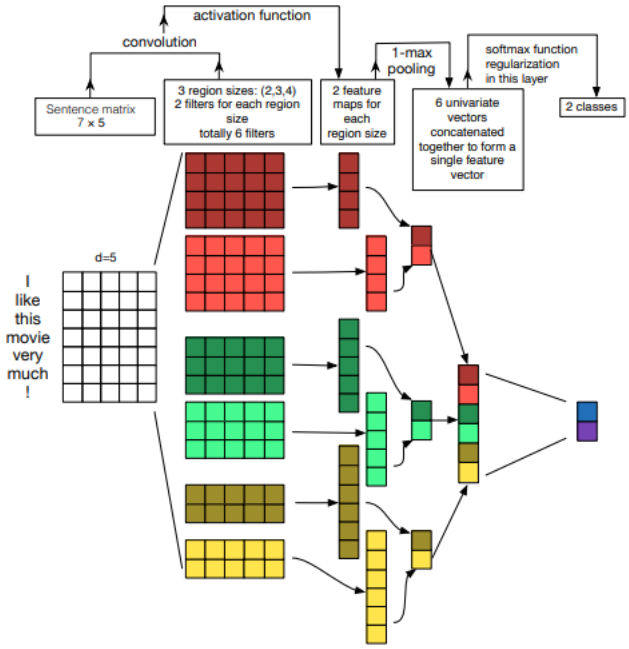


FIGURE 2.4: Shallow-and-wide CNN, *image from Zhang and Wallace, (2015)*.

All of these works are based on word input tokens, following Bengio, Ducharme, and Vincent, (2000), which introduced for the first time a solution to fight the curse of dimensionality thanks to distributed representations. A limit of this approach is that typical sentences and paragraphs contain a small number of words, which prevents the previous convolutional models to be very deep: most of them indeed

only have two layers. Other works (Santos and Gatti, 2014) further noted that word-based input representations may not be very well adapted to social media inputs like Twitter, where tokens usage may be extremely creative: slang, elongated words, contiguous sequences of exclamation marks, abbreviations, hashtags,... Therefore, they introduced a convolutional operator (Figure 2.5) on characters to automatically learn the notions of words and sentences. This enables neural networks to be trained end-to-end on texts without any pre-processing, not even tokenization.

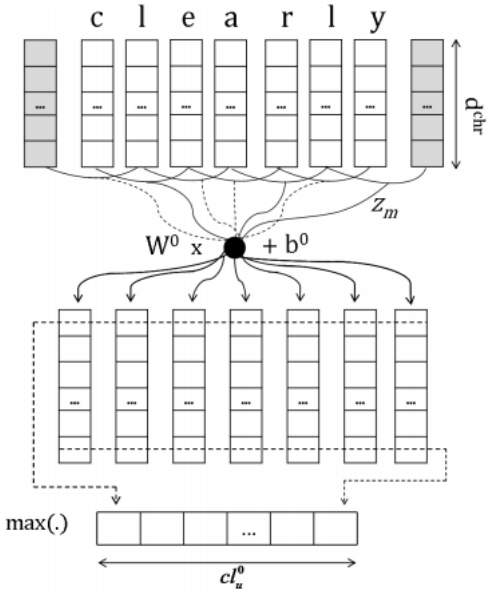


FIGURE 2.5: Convolutional approach to character-level feature extraction. *Credit image: (Santos and Gatti, 2014).*

Later, Zhang, Zhao, and LeCun, (2015) enhanced this approach and proposed a deep CNN for text: the number of characters in a sentence or paragraph being much longer, they can train for the first time up to 6 convolutional layers. However, the structure of this model (Figure 2.6) is designed by hand by experts and it is thus difficult to extend or generalize the model with arbitrarily different kernels and pool sizes. Hence, Schwenk et al., (2017), inspired by He et al., (2016), presented a much simpler but very deep model with 29 convolutional layers.

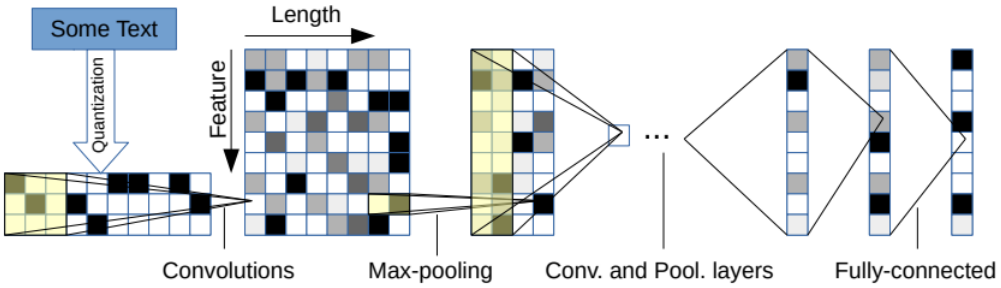


FIGURE 2.6: Character-level with 6 Convolutional Layers Network for Text Classification. *Credit image: (Zhang, Zhao, and LeCun, 2015).*

Subsequently, there has been numerous efforts to build a unique model that is capable of capturing global dependency by RNN while still being well aware of local patterns by CNN for NLP and not just for text classification. Kim et al., (2016) introduced a character aware neural language model by combining a CNN on character

embeddings with a highway LSTM on subsequent layers. Radford, Jozefowicz, and Sutskever, (2017) also explored a multiplicative LSTM (mLSTM) on character embeddings and found that a basic logistic regression learned on this representation can achieve state-of-the-art result on the Sentiment Tree Bank dataset (Socher et al., 2013) with only a few hundred labeled examples. Capitalizing on the effectiveness of character embeddings, Dhingra et al., (2017) proposed a hybrid word-character models to leverage the avantages of both worlds. However, their initial experiments show that this simple hybridation does not bring very good results: the learned representations of frequent and rare tokens of words and characters is different and co-training them may be harmful. To alleviate this issue, Miyamoto and Cho, (2016) proposed a scalar gate to control the ratio of both representations, but empiricial studies showed that this fixed gate may lead to suboptimal results. Yang, Salakhutdinov, and Cohen, (2016) then introduced a fine-grained gating mechanism to combine both representations. They showed improved performance on reading comprehension datasets, including Children's Book Test and SQuAD. Another different but popular way to achieve open-vocabulary (rare and unknown words) capability is to use preprocessing byte pair encoding compression algorithm (Sennrich, Haddow, and Birch, 2016). This allows the model to cope better with morphological changes and learn compounding and transliteration via subword units.

Neural models are also applied to various problems of Sentiment Analysis: document level and aspect level sentiment classification, sentiment composition, temporal opinion mining, sarcasm analysis, emotion analysis,... It is now considered as de facto standard technique to study and produce state-of-the-art results on these tasks. We refer the interested readers to the comprehensive review in (Zhang, Wang, and Liu, 2018) for more details.

2.2 Sentiment analysis in dialogues

Although being popular, the neural model is still not well studied for sentiment analysis in the context of dialogues, although there exists close relations and strong respective influences between these two aspects. This motivates us to fill this gap by extending our researches towards interactions between sentiment and dialogue. We start by defining the basic concepts of dialog acts that we will use next.

Dialogue act Dialog acts (DA), a type of speech act (Austin, 1962), is an utterance that serves a function in the dialog in broad terms. A typical dialog system consists of a taxonomy of classifying different functions dialog acts can play. Stolcke et al., (2000) introduced more than 40 tags of different dialog acts often seen in conversational speech. Examples of types in this study are shown in the second column of Table 2.7 where each utterance is assigned a unique DA label in a conversation which involves two speakers talking about one of several general topics.

The ability to model and automatically recognize dialog act is an important step towards understanding spoken language. Thus, the problem received a lot of attentions from the scientific community. Traditionally, dialog act tagging has followed a supervised approach, which starts with the developing of annotation guidelines, labeling of corpora and training a tagger to classify dialog acts.

Stolcke et al., (2000) developped a probabilistic integration of speech recognition with dialog modeling based on a Hidden Markov model (HMM), in which lexical, prosodic and acoustic features are employed to improve both speech and dialog act classification accuracy. Forsythand and Martell, (2007) studied and built a chat

Speaker	Dialogue Act	Utterance
A	YES-NO-QUESTION	So do you go to college right now?
A	ABANDONED	Are yo-,
B	YES-ANSWER	*Yeah,*
B	STATEMENT	*it's my last year [laughter].*
A	DECLARATIVE-QUESTION	You're a, so you're a senior now.
B	YES-ANSWER	*Yeah,*
B	STATEMENT	*I'm working on my projects trying to graduate [laughter].*
A	APPRECIATION	Oh, good for you.
B	BACKCHANNEL	*Yeah.*
A	APPRECIATION	That's great,
A	YES-NO-QUESTION	um, is, is N C University is that, uh, State,
B	STATEMENT	*N C State.*
A	SIGNAL-NON-UNDERSTANDING	What did you say?
B	STATEMENT	*N C State.*

FIGURE 2.7: Fragment of a labeled conversation (from the Switchboard corpus of human-human conversational telephone speech) *Credit image: (Stolcke et al., 2000).*

corpus with a higher complexity than a normal spoken dialog when multiple topics are being discussed by multiple people simultaneously. Jeong, Lin, and Lee, (2009) proposed a semi-supervised learning approach to transfer dialog acts from labeled speech corpus to forums and e-mail. They attempted to create domain-independent acts with the restructure of the source act inventories. Vosoughi and Roy, (2016) created six speech acts for Twitter and applied a multi-class classification problem to discover dialog act based on a set of semantic and syntactic features.

This annotate-train-test paradigm has been succesful but with a cost of expensive labeling process and limiting the amount of training data. Thus, Woszczyna and Waibel, (1994) proposed an unsupervised HMM for a dialog structure of meeting scheduling. Crook, Granell, and Pulman, (2009) employed Dirichlet mixture models to group utterances into a number of common acts and ingored the dialogue sequential structure. Ritter, Cherry, and Dolan, (2010) presented various unsupervised approach to cluster raw utterances and infer their corresponding dialog acts from noisy conversations on Twitter.

Sentiment Analysis and Dialogue act Pluwak, (2016) demonstrates that some expressions of sentiments might not be detected with traditional methods of opinion mining, and that exploiting dialog acts may partly solve this challenge. To provide an example in the sport domain: *"Bring the old goal-keeper back !"*, does not convey a direct polarization of sentiment but through an inference of the act of demand and advice: 1) the player and fans' unhappiness with the current one and 2) preference for the previous player - both are indicators of a negative judgement towards this goal-keeper. Using such an analysis of the function of each utterance allows the right interpretation of such statements. Therefore, there is a need to understand opinions and expressions of attitude in a broader sense than sentiment recognition based on lexicons or parts-of-speech. Indeed, researchers have long time seen sentiment analysis and dialog act recognition in an intimately close relationship. Clavel and Callejas, (2016) studied the impact of both dialog act recognition and sentiment analysis in human-agent conversational platforms and affective conversational interfaces. An embodied conversational agent (ECA) (Figure 2.8), virtual assistant interacting with humans, has to consider the human emotional behaviors and attitudes in order to adapt its behavior accordingly.

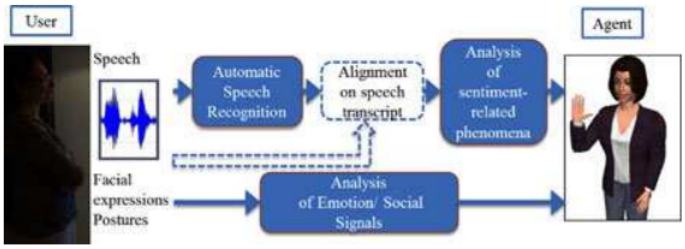


FIGURE 2.8: Integrating sentiment analysis in a multimodal context in a human-agent interaction. *Credit image: (Clavel and Callejas, 2016).*

Further, emotion detection can be used to help a dialog act classifier. Boyer et al., (2011) exploited affects that are visually expressed on faces to better classify dialog acts. Novielli and Strapparava, (2013) performed a dialog act clustering of a lexicon and studied the emotional load of dialog acts. The authors further also tried to improve dialog act recognition by exploiting affective lexicon with encouraging results.

Conversely, dialog acts can also help the emotion detection task. Hasegawa et al., (2013) showed how to predict emotions in online dialogues. They considered a history, which is composed of an utterance and a response to it (Figure 2.9), and trained a system to predict the addressee's emotion caused by the response. In a slightly similar spirit, Herzig et al., (2016) considered the full context of a longer dialogue and include dialog features (e.g., time elapsed between turns...) into an emotion classifier.

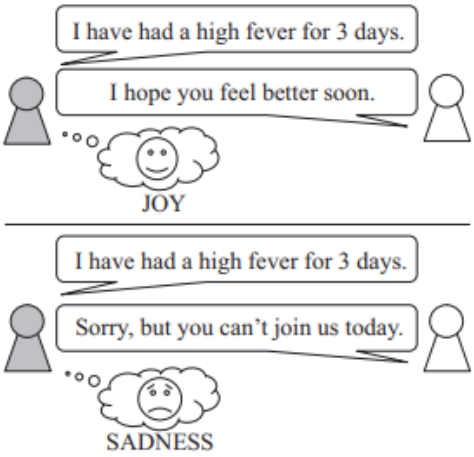


FIGURE 2.9: Two example pairs of utterances and responses which elicit certain emotions, JOY or SADNESS, in the addressee's mind. *Credit image: (Hasegawa et al., 2013).*

Still, there is little work combining both SA and DA for social media. A rare work that we know is from Kim and Kim, (2018), who proposed a joint model of sentiments and dialog acts on a Korean corpus. Their proposed Integrated Intention Identification Model (IIIM) is shown in Figure 2.10, where they exploited one convolutional network per each task of speech acts, predicators, and sentiments. The output vectors are then concatenated and passed to one classifier per task.

To learn the embedding vectors, three cycles of partial error backpropagations are applied. The errors between the output and the gold label of speech act (and consecutively predicator, sentiment) are propagated through the partially connected

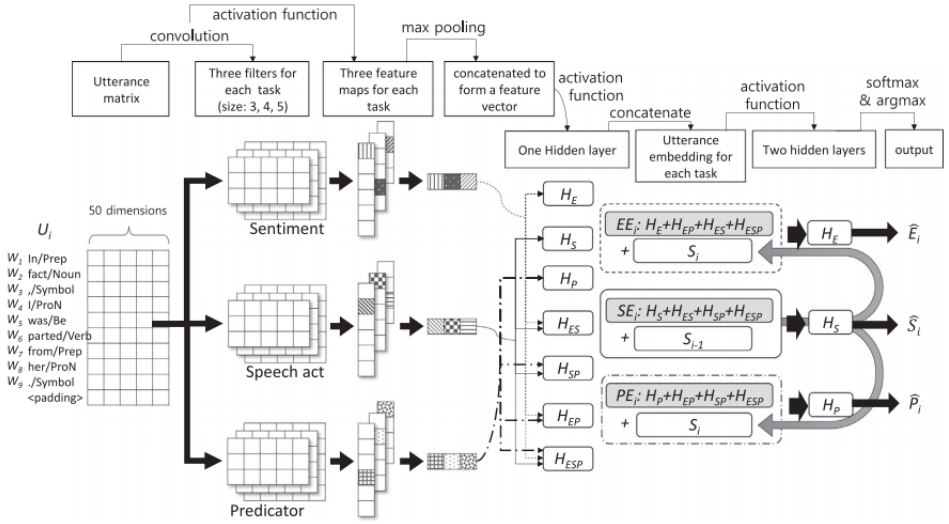


FIGURE 2.10: Overall architecture of IIIM. *Credit image: (Kim and Kim, 2018).*

nodes. The embedding vectors are expected to accumulate informative features thanks to these successive partial error backpropagations. With this learning paradigm and the joint model design, authors showed better performance than each independent models.

Chapter 3

Impact of Neural Networks depth for sentiment analysis

3.1 Introduction

Following the success of deep learning approaches in the ImageNet competition, there has been a surge of interest in the application of deep models for text classification and sentiment analysis, with deeper and deeper networks being proposed in the literature: (Kim, 2014) (shallow-and-wide CNN layer), (Zhang, Zhao, and LeCun, 2015) (6 CNN layers), (Schwenk et al., 2017) (29 CNN layers). Besides the development of deep networks, there is a debate about which atom-level (word or character) would be the most effective for Natural Language Processing (NLP) tasks. Word embeddings, which are continuous representations of words, initially proposed by Bengio, Ducharme, and Vincent, (2000) and widely adopted after word2vec (Mikolov et al., 2013) have been chosen as the main standard representations for most NLP tasks. Based on this representation, a common belief is that, similarly to vision, the model will learn hiearchical features from the text: words combine to form n-grams, phrases, sentences... Nevertheless, it is still not clear which atom-level is the best and whether very deep networks at the word-level are really better for text classification.

Motivated by these questions, we propose an adaptation of a DenseNet (Huang et al., 2016) (Section 3.3) for text classification and sentiment analysis at the word-level, which we compare with the state-of-the-art. We bring elements of a response by providing a full comparison of a shallow-and-wide CNN (Kim, 2014) both at the character and word levels on the 5 datasets described in Zhang, Zhao, and LeCun, (2015) (Section 3.4). Besides, we also show the benefit of a deep neural network on Twitter financial data with transfer learning via a distant supervision approach in Section 3.2.

3.2 Preliminary analysis of sentiment transfer by a Deep CNN

3.2.1 Motivation

Recent successes of deep learning methods go along with a strong dependence on massive labeled data set (Krizhevsky, Sutskever, and Hinton, 2012; Hinton et al., 2012). In the meanwhile, there are a lot of domains where the collection of labeled data is difficult or simply very expensive. Twitter is such a case. Despites of being abundant of data, human can never label all new tweets posted every day. *Distant supervision*, a method which *automatically* collects samples with some pre-defined rules from a larger pool of data and transfers them to another smaller similar task, can tackle this issue. Effectively, on Twitter, Go, Bhayani, and Huang, (2009) showed that emoticons can be a reliable and effective clue for such rules for the sentiment

analysis task. Tang et al., (2014a) and Deriu et al., (2016) have followed this stategy and ended up winning Semeval competitions. In these works, they only transfered a shallow convolutional networks (Kim, 2014) on word-level and achieved already remarkable result.

Moreover, neural networks are extremely good at learning the input representation only when the network is deep with abundant data (Yosinski et al., 2014). In previous works, Zhang, Zhao, and LeCun, (2015) and Schwenk et al., (2017) have shown that it is possible to build such a deep network for text classification task with character input level. In the context of Twitter, this is more convenient as it allows the model to learn special tokens, slangs, elongated words, contiguous sequences of exclamation marks, abbreviations, hashtags,... Motivated by these observations, we explore the capability of transferring knowledge learning from abundant data of Twitter to Semeval dataset with a deep convolutional network. The benefit of a deep structure for neural networks is demonstrated in many domains of computer vision and speech recognition but there is no such similar firm claim on natural language processing task until now. To investigate this question, we explore the structure of deep models on character-level as described in Conneau et al., (2016). However, as the length of the sequence in twitter is shorter than in the amazon and movie reviews used in Conneau et al., (2016), we don't perform pooling operation and use two models: a small model with (1-1-1) convolutional blocks and a bigger model with (2-2-1) convolutional blocks using (256-256-128) features in each block correspondingly. Following Conneau et al., (2016), each convolutional block contains two consecutive convolutional layers, where each one followed by a temporal BatchNorm layer (Ioffe and Szegedy, 2015) and an ReLU activation. For the last fully connected layers, we use (5632-4096-2048-512-128) features respectively to observe better the transfer's process. The rest of other hyperparameters follow Conneau et al., (2016).

3.2.2 Twitter data

Stanford140 Emoticon Twitter data

The first dataset that we used for the study consists of 1.6 millions tweets, automatically collected in 2009 on a wide range of topics with the Twitter Search API (Go, Bhayani, and Huang, 2009). This corpus is completed with 800k tweets containing either the positive or negative emoticons ":)", ":(". The training dataset is then preprocessed:

- Emoticons are stripped off;
- Any tweet containing both positive and negative emoticons is removed;
- Retweets, duplicated tweets and tweets with ":P" are also removed to reduce as much as possible the bias in the corpus.

Further details of the preprocessing we applied can be followed in Go, Bhayani, and Huang, (2009). Besides, to supplement this corpus with more recent data, we collected 343k positive and negative tweets with the same emoticon rules via Twitter Streaming from January to March 2017. The characteristics of the resulting corpus are shown in Figure 3.1 and 3.2. There are 67 characters on average in each tweets; 239 tweets exceeding the maximum limit of 176 characters lengths are truncated. Most of these tokens are characters and numbers; there is only a small number of punctuations. There is a concern that if the limit size of the sequence is bigger than

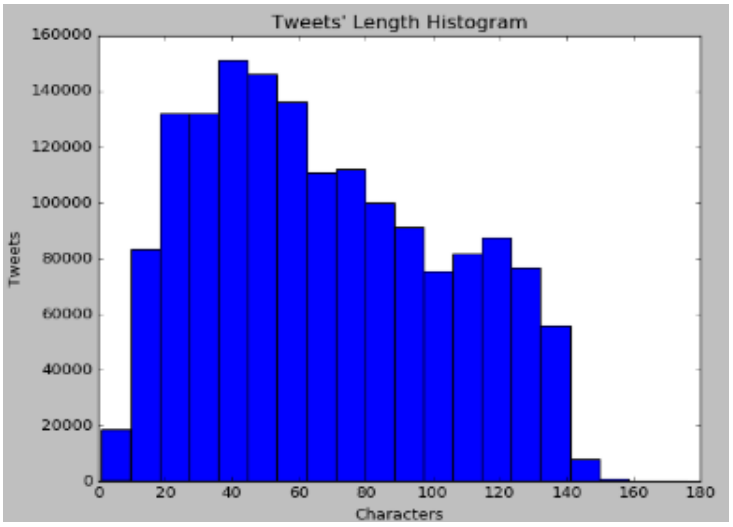


FIGURE 3.1: Length distribution of the Stanford Emoticon Twitter data

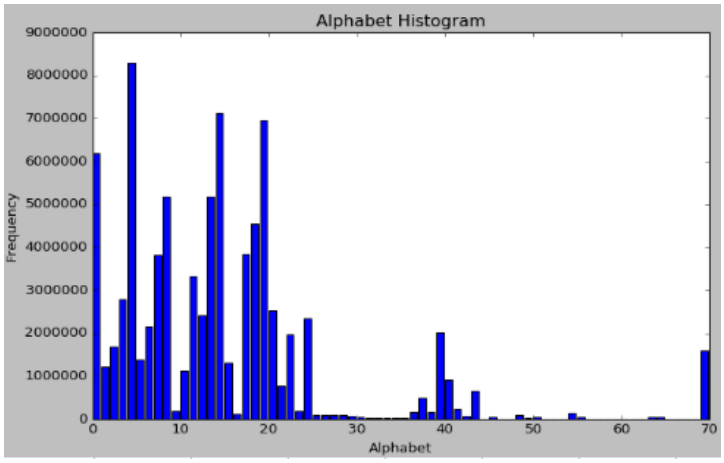


FIGURE 3.2: Vocabulary character distribution of the Stanford Emoticon Twitter data

two times of the mean or median, the result of the convolutional model will be less precise because there are many useless padding 0s on the right side of the sentence representation. We thus examined a maximum sequence length of 140 and 160, but such effect has not been seen in this data. We first trained a model with the configuration of (1-1-1) double convolutional blocks on the Stanford140 dataset. It took 14k steps to converge and the test accuracy is 80%. Using a bigger model (2-2-1) leads to a gain of 5% of accuracy on the test set.

SESAMm data

Stanford140 is still a small dataset as comparing to the potential quantity of data one can obtain from Twitter. SESAMm [1] generously provided us with 3 billion subjective tweets that they collected automatically from the period 2014-2016. These tweets do not contain any particular filter of preprocessing emoticons. Nevertheless, they defined the subjectivity crtierion by selecting the tweets containing "I feel, I am, I

[1] *http://www.sesamm.com/*

think,...". From this pool, we filtered out 16 millions balanced positive/negative tweets with the rules of Go, Bhayani, and Huang, (2009) for the sake of comparison. Surprisingly, the model trained on this dataset gave the same 85% of accuracy on this test set.

3.2.3 Transfer learning results

We carried out transfer learning experiments from the two datasets that have been described, i.e., the augmented Stanford140 (see Section 3.2.2) and SESAMm data (Section 3.2.2), to the target Semeval 2013 dataset, subtask Message Polarity Classification (Nakov et al., 2013). Semeval 2013 dataset is essentially composed of tweets in training and and a mix of tweets-sms in test set with three labels: positive, negative, and neutral sentiment. To assure a well discriminated classifier, we first check the case of only positive and negative classes. After training our model on the augmented Stanford140 corpus, fine-tuning the last layer of deep convolutional network on the Semeval training set resulted in a 75% Semeval test accuracy performance. Fine-tuning the 2 or 3 last fully connected layers degrade the performance. This is consistent with the conclusion of Yosinski et al., (2014): when the target transfer data is small and the domain is close, one should reuse the whole network structure and re-train only the last linear classifier. For the case of SESAMm data, a clear improvement is observed with a performance of 84% on the Semeval test set.

Learning on 3 classes. Besides learning on two sentiment polarities, we also evaluated on three classes, with the neutral class like in (Nakov et al., 2013). We selected 800k tweets without emoticons from SESAMm as a subcorpus of neutral tweets and added it in the Stanford140 dataset. Although this is only a crude approximation to obtain neutral tweets, it was shown to work quite well in the literature (Go, Bhayani, and Huang, 2009; Tang et al., 2014a). The potential impact of this loose approximation for neutral samples is expected to become less important when the quantity of data included becomes larger. In terms of performances, we obtained with a model trained on this corpus only 55% in F_1^{PN} (average of F1-scores for the positive and negative classes), which is below the value of 69.02% of the state-of-the-art results (Nakov et al., 2013). This result confirms that considering neutral samples makes sentiment analysis more challenging. Also, this poor result may be explained by the bad choice of neutral tweets from SESAMm data, which contains only subjective tweets and is thus far from the Semeval test distribution.

Learning on 9 classes. To get away from the difficulty of defining and collecting neutral samples, we exploited a broader set of sentiment classes, composed of 9 common classes corresponding to their respective emoticons: *happy, laughing, kiss-wink, playful, sad, horror, shock, annoyed* and *hesitated*, are extracted on the SESAMm dataset. The icons and their corresponding classes are shown in Table 3.1. We used this finer-grained set of emotions in order to help the classifier to discriminate better and eventually generalize better. It turned out that transferring such a model to 3 Semeval classes still gave the same level of performances than before. It is hard to conclude whether the emoticon filter on the SESAMm dataset is noisy and thus not really helpful, or whether real hand-labeled samples are required to obtain better models.

Icons	**Classes**
:-) :) :-] :] :-3 :3 :-> :> 8-) 8) :-} :} :o) :c) =] =) :-)) :'-) :')	Happy
:-D :D 8-D 8D x-D xD X-D XD =D =3	Laughing
:-* :* :x ;-) ;) *-) *) ;-] ;] :-, ;D	Kisswink
:-P :P X-P XP x-p xp :-p :p :-b :b d: =p >:P	Playful
:-(:(:-c :c :-< :< :-[:[:-| | >:[:{ :@ >:(:'-(:'(	Sad
D-': D:< D: D8 D; D= DX	Horror
:-O :O :-o :o :-o 8-0 >:O	Shock
:-/ :/ :-. >: >:/ : =/ = :L =L :S	Annoyed

TABLE 3.1: The icons are chosen from Wikipedia's list: https://en.wikipedia.org/wiki/List_of_emoticons

3.2.4 Conclusion

Deep Learning successes are, at least originally, explained by a lot of labeled data and a very deep structure design of the neural network. Interestingly, these two properties are along crucial elements to enable transfer learning to low-resource test domain, as shown in Yosinski et al., (2014) in the computer vision domain. In our preliminary study, we have investigated the transfer learning ability of a deep network from Twitter data, where unlabled data is abundant, to the Semeval 2013 dataset, for sentiment analysis task. As Twitter often contains special tokens, we have studed character-level inputs. We have presented an adaptation of a deep convolutional network (Conneau et al., 2016) for social media data and compared it with state-of-the-art result based on shallow-and-wide convolutional neural networks. Transfer learning results from SESAMm data showed that deep network performs worse than the state-of-the-art result on three classes sentiment prediction. We suspect that this is both due to the difficulty of data construction of the transfer learning source data and the challenge of transfer learning on natural language data, which is essentially different than in the image processing domain.

3.3 DenseNet

In order to study the potential importance of depth of neural networks for text classification, we propose a model that is inspired by the DensetNet model from Huang et al., (2016), which has been shown to give very good results on image classification. We adapt this model for the text classification task with the following components.

3.3.1 Skip-connections

In order to enable training of deep convolutional models, a skip-connection (He et al., 2016) is often used to modify the non-linear transformation $\mathbf{x}_l = \mathcal{F}_l(\mathbf{x}_{l-1})$ between the output activations $\mathbf{x}_{l-1}$ at layer $l-1$ and at layer l with an identity function:

$$\mathbf{x}_l = \mathcal{F}_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1} \tag{3.1}$$

This allows the gradient to backpropagate deeper in the network and limits the impact of various issues such as vanishing gradients.

3.3.2 Dense Connectivity

However, as suggested in the literature, the additive combination of this skip connection with $\mathcal{F}_l(\mathbf{x}_{l-1})$ may negatively affect the information flow in the model; we

thus actually employ an alternative concatenation operator (Huang et al., 2016), which allows to create direct connections from any layer to all subsequent layers. Hence, the l^{th} layer has access to the feature maps of all preceding layers, $\mathbf{x}_0, \cdots, \mathbf{x}_{l-1}$, as input:

$$\mathbf{x}_l = \mathcal{F}_l([\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{l-1}]) \tag{3.2}$$

This model, DenseNet, can be viewed as an extreme case of a ResNet (He et al., 2016). The distance between both ends of the network is shrinked and the gradient may backpropagate more easily from the output back to the input. Figure 3.3 illustrates the idea where multiple convolutional filters output 2D matrices are all concatenated together before going into another dense block.

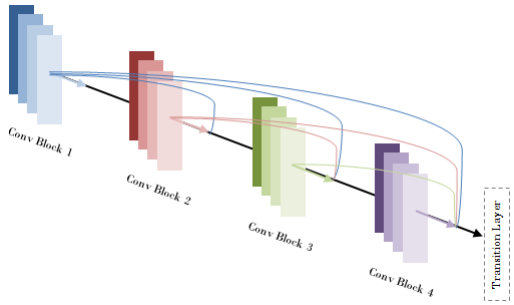


FIGURE 3.3: Dense Block.

3.3.3 Convolutional Block and Transitional Layer

Following He et al., (2016), we define $\mathcal{F}_l(.)$ as a function of three consecutive operations: batch normalization (BN), rectified linear unit (ReLU) and a 1x3 convolution. To adapt the variability of the changing dimension of the concatenation operation, we define a transition layer which composes a 1x3 convolution and a 1x2 local max-pooling between two dense blocks. Given a vector c^{l-1} outputed by a convolutional layer $l-1$, the local max-pooling layer l outputs a vector c^l:

$$\left[c^l\right]_j = \max_i \left[c^{l-1}\right]_{k\times(j-1)\leq i<k\times j} \tag{3.3}$$

where $1 \leq i \leq n$ and k is the kernel pooling size, n is the sequence length. The word-level DenseNet model is the same as the character-level model shown in Figure 3.4, except for the last two layers, where the local max-pooling and the two fully connected layers are replaced by a single global average pooling layer. In the figure, **3, Temp Conv, 128** means temporal convolutional operation with kernel window size = 3 and filter size = 64; **pool/2** means local max-pooling with kernel size = stride size = 2, it will reduce the size of the sequence by a half. We empirically observed that better results are obtained with word tokens.

3.4 Evaluation

Task and data We test our models on the 5 datasets used in Zhang, Zhao, and LeCun, (2015) and summarized in Table 3.2, where the tasks include **ENC**: English News Categorization, **SA**: Sentiment Analysis, **OC**: Ontology Classification, **TC**: Topic Classification. The datasets are:

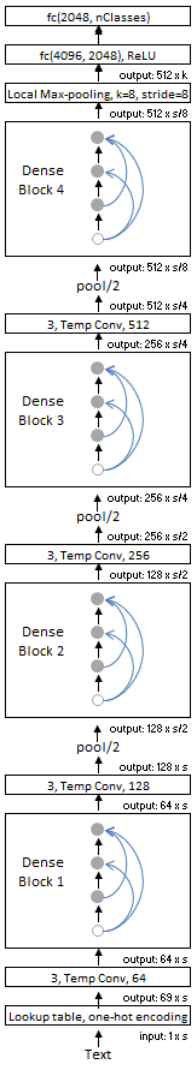


FIGURE 3.4: DenseNet model for Text classification.

- AGNews: internet news articles AGNews composed of titles plus descriptions and classified into 4 categories: World, Entertainment, Sports and Business, with 30k training samples and 1.9k test samples per class.
- Yelp Review Polarity: The Yelp review dataset is obtained from the Yelp Dataset Challenge in 2015. Each polarity dataset has 280k training samples and 19k test samples.
- Yelp Review Full: The Yelp review dataset is obtained from the Yelp Dataset Challenge in 2015. It has four polarity star labels: 1 and 2 as negative, and 3 and 4 as positive. Each star label has 130k training samples and 10k testing samples.
- DBPedia: DBPedia is a 14 non-overlapping classes picked from DBpedia 2014 (wikipedia). Each class has 40k training samples and 5k testing samples.
- Yahoo! Answers: ten largest main categories from Yahoo! Answers Comprehensive Questions and Answers version 1.0. Each class contains 140k training samples and 5k testing samples, including question title, question content and

best answer. For DenseNet on word-level, we only used 560k samples because of lack of memory.

Dataset	#y	#train	#test	Task
AGNews	4	120k	7.6k	ENC
Yelp Binary	2	560k	38k	SA
Yelp Full	5	650k	38k	SA
DBPedia	14	560k	70k	OC
Yahoo	10	1 400k	60k	TC

TABLE 3.2: Statistics of datasets used in our experiments.

Hyperparameters and Training For all experiments, we train our model's parameters with the Adam Optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001, a mini-batch size of 128. The model is implemented using Tensorflow and is trained on a GPU cluster (with 12Gb RAM on GPU). The hyperparameters are chosen following Zhang, Zhao, and LeCun, (2015) and Kim, (2014), which are described below. On average, it takes about 10 epochs to converge.

Character-level Following Zhang, Zhao, and LeCun, (2015), each character is represented as a one-hot encoding vector where the dictionary contains the following 69 tokens: *"abcdefghijklmnopqrstuvwxyz*0123456789−, ; .!? :' "/|_#%& ˆ*˜+ =<> ()[]". The maximum sequence length is 1014; smaller texts are padded with 0 while larger texts are truncated. The convolutional layers are initialized following Glorot and Bengio, (2010).

Word-level The embedding matrix W is initialized randomly with the uniform distribution between $[-0.1; 0.1]$ and is updated during model's training using back-propagation. The embedding vectors have 300 dimensions and are initialized with word2vec vectors pretrained on 100 billion words from Google News (Mikolov et al., 2013). Out-of-vocabulary words are initialized randomly. A dropout of 0.5 is used to prevent overfitting. The kernel window sizes for character tokens are $N_f = (15, 20, 25)$ with $m = 700$ filters. For word-level, $N_f = (3, 4, 5)$ with $m = 100$ filters. The shallow-and-wide CNN requires 10 hours of training on the smallest dataset, and up to one day on the largest. The DenseNet model requires 2 to 4 days for training.

Results Table 3.3 details the accuracy obtained with our models (10 rows on top) and compare them with state-of-the-art results (13 rows at the bottom) on 5 corpus and text classification tasks (columns), where N_{big} and N_{Xbig} correspond to the configuration number of blocks $N_b = (4 - 4 - 4 - 4)$ and $N_b = (10 - 10 - 4 - 4)$ respectively; **GAP** means Global Average-Pooling and **LMP** means Local Max-Pooling.

The models from the litterature we compare to are:

- **bag of words:** The BOW model exploits the most frequent words from the training data (Zhang, Zhao, and LeCun, 2015)
- **ngrams:** The bag-of-ngrams model exploits the most frequent word n-grams from the training data (Zhang, Zhao, and LeCun, 2015)

Models	AGNews	Yelp Bin	Yelp Full	DBPedia	Yahoo
Char shallow-and-wide CNN	90.7	94.4	60.3	98.0	70.2
Char-DenseNet N_{big} GAP	90.4	94.2	61.1	97.7	68.8
Char-DenseNet N_{Xbig} GAP	90.6	94.9	62.1	98.2	70.5
Char-DenseNet N_{big} LMP	90.5	95.0	63.6	98.5	72.9
Char-DenseNet N_{Xbig} LMP	92.1	95.0	64.1	98.5	73.4
Word shallow-and-wide CNN	92.2	**95.9**	**64.9**	**98.7**	73.0
Word-DenseNet N_{big} GAP	91.7	95.8	64.5	**98.7**	70.4*
Word-DenseNet N_{Xbig} GAP	91.4	95.5	63.6	98.6	70.2*
Word-DenseNet N_{big} LMP	90.9	95.4	63.0	98.0	67.6*
Word-DenseNet N_{Xbig} LMP	88.8	95.0	62.2	97.3	68.4*
bag of words	88.8	92.2	58.0	96.6	68.9
ngrams	92.0	95.6	56.3	98.6	68.5
ngrams	92.4	95.4	54.8	**98.7**	68.5
fastText	**92.5**	95.7	63.9	98.6	72.3
char-CNN	87.2	94.7	62.0	98.3	71.2
char-CRNN	91.4	94.5	61.8	98.6	71.7
very deep char-CNN	91.3	95.7	64.7	**98.7**	73.4
Naive Bayes	90.0	86.0	51.4	96.0	68.7
Kneser-Ney Bayes	89.3	81.8	41.7	95.4	69.3
MLP Naive Bayes	89.9	73.6	40.4	87.2	60.6
Discriminative LSTM	92.1	92.6	59.6	**98.7**	**73.7**
Generative LSTM-indep. comp.	90.7	90.0	51.9	94.8	70.5
Generative LSTM-shared comp.	90.6	88.2	52.7	95.4	69.3

TABLE 3.3: Accuracy of our proposed models and of state-of-the-art models from the litterature.

- **ngrams TFIDF:** Same as the ngrams model but uses the words TFIDF (term-frequency inverse-document-frequency) as features (Zhang, Zhao, and LeCun, 2015)
- **fastText:** A linear word-level model with a rank constraint and fast loss approximation (Joulin et al., 2016)
- **char-CNN:** Character-level Convolutional Network with 6 *hand-designed* CNN layers (Zhang, Zhao, and LeCun, 2015)
- **char-CRNN:** Recurrent Layer added on top of a Character Convolutional Network (Xiao and Cho, 2016)
- **very deep CNN:** Character-level model with 29 Convolutional Layers inspired by ResNet (Schwenk et al., 2017)
- **Naive Bayes:** A simple count-based word unigram language model based on the Naive Bayes assumption (Yogatama et al., 2017)
- **Kneser-Ney Bayes:** A more sophisticated word count-based language model that uses tri-grams and Kneser-Ney smoothing (Yogatama et al., 2017)
- **MLP Naive Bayes:** An extension of the Naive Bayes word-level baseline using a two layer feedforward neural network (Yogatama et al., 2017)
- **Discriminative LSTM:** Word-level model with logistic regression on top of a traditional LSTM (Yogatama et al., 2017)

- **Generative LSTM-independent comp.:** A class-based word language model with no shared parameters across classes (Yogatama et al., 2017)
- **Generative LSTM-shared comp.:** A class-based word language model with shared components across classes (Yogatama et al., 2017)

Figure 3.5 visually compares the performances of 3 character-level models with 2 word-level models. Character-level models include our shallow-and-wide CNN model, as well as two models from the literature: a CNN with 6 layers from Zhang, Zhao, and LeCun, (2015), and another CNN with 29 layers from Schwenk et al., (2017).

For word-level models, we present our shallow-and-wide CNN with respect to the best DenseNet with number of blocks $N_b = (4-4-4-4)$ using Global Average-Pooling.

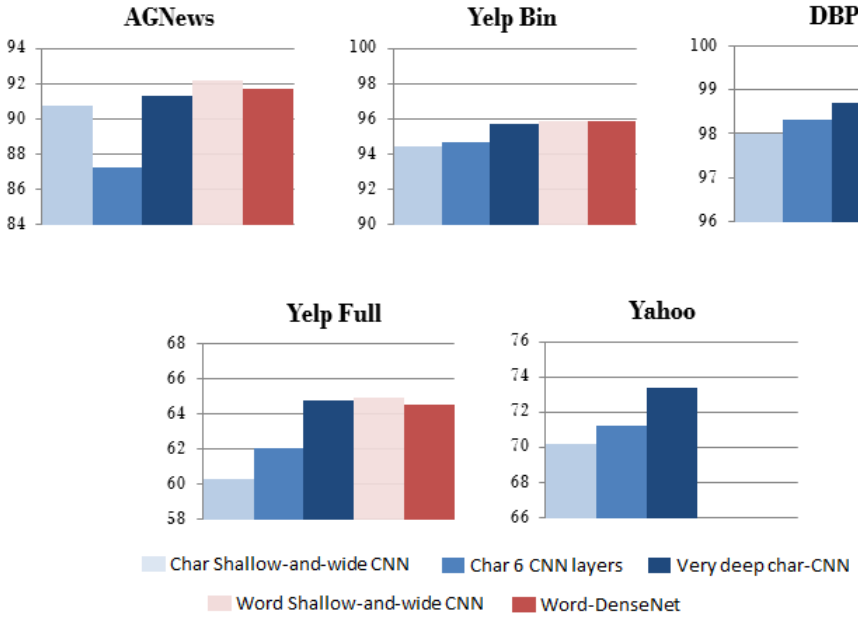


FIGURE 3.5: Comparison of character (*in blue, on the left*) and word-level (*in red, on the right*) models on all datasets.

The main conclusions of these experiments can be summarized into the three following paragraphs.

Impact of depth for character-level models

Deep character-level models do not significantly outperform the shallow-and-wide network. A shallow-and-wide network (row 1 in Table 3.3) achieves 90.7%, 94.4%, 98.0% on AGNews, Yelp Bin, DBPedia respectively, comparing to 91.3%, 95.7%, 98.7% of a very deep CNN (Schwenk et al., 2017). Although the deep structure achieves a slight gain in performance on these three datasets, the difference is not significant. Interestingly, a very simple shallow-and-wide CNN can get very close results to the deep 6 CNN layers of (Zhang, Zhao, and LeCun, 2015) which structure must be designed meticulously. For the smallest dataset AGNews, we suspect that the deep model **char-CNN** performs badly because it needs more data to take benefit from depth. The deep structure gives an improvement of about 4% on Yelp Full and Yahoo (first row of Table 3.3 vs. **very deep char CNN**), which is interesting but does not match the gains observed in image classification. We have tried various

configurations: $N_f = (15, 20, 25)$, $N_f = (10, 15, 20, 25)$ and $N_f = (15, 22, 29, 36)$ on shallow models but they didn't do better.

Impact of depth for word-level models

The DenseNet is better with 20 layers $N_b = (4 - 4 - 4 - 4)$ than with 32 layers $N_b = (10 - 10 - 4 - 4)$ and Global Average-Pooling is better than the traditional Local Max-pooling. It is the opposite for char-level. This is likely a consequence of the fact that the observed sequence length is much shorted with words than with characters. However, the main striking observation is that all deep models are matched or outperformed by the shallow-and-wide model on all datasets, although it is still unclear whether this is because the input sequences are too short to benefit from depth or for another reason. Further experiments are required to investigate the underlying reasons of this failure of depth at word-level.

State-of-the-art performances with shallow-and-wide word-level model

With a shallow-and-wide network on word-level, we achieved a very close result to the state-of-the-art on AGNews, obtained state-of-the-art results on 2 datasets: DBPedia, Yahoo, and set new state-of-the-art levels on two datasets: Yelp Binary and Yelp Full. We also empirically found that a word-level shallow model may outperform a very deep char-level network. This confirms that word observations are still more effective than character inputs for text classification. In practice, quick training on word-level with a simple convolutional model may already produce good result.

3.5 Discussion

Text representation - discrete, sparse

Very deep models do not seem to bring a significant advantage over shallow networks for text classification, as opposed to their performances in other domains such as image processing. We believe one possible reason may be related to the fact that images are represented as real and dense values, as opposed to discrete, artificial and sparse representations for text. The first convolutional operation results in a matrix (2D) for image while it results in a vector (1D) for text. The same deep network applied to two different representations (dense and sparse) will obviously get different results. Empirically, we found that a deep network on 1D (text) is less effective and learns less information than on 2D (image).

Local vs Global max-pooling

A global max-pooling (Collobert and Weston, 2008), which retrieves the most influencial feature could already be good enough for sparse and discrete input text, and gives similar results than a local max-pooling with a deep network.

Word vs Character level

Character-level models could be a choice but word-level is still the most effective method. Moreover, in order to use char-level representation, we must use a very deep model, which is less practical because it takes a long time to train.

3.6 Conclusion

In computer vision, several works have shown the importance of depth in neural networks and the major benefits that can be gained by stacking many well-designed convolutional layers in terms of performances. However, such advantages do not necessarily transfer to other domains, and in particular Natural Language Processing, where the impact of depth in the model is still unclear. This chapter exploits a number of additional experiments to further explore this question and potentially bring some new insights or confirm previous findings. We further investigate another related question about which type of textual inputs, characters or words, should be chosen at a given depth. By evaluating on several text classification and sentiment analysis tasks, we show that a shallow-and-wide convolutional neural network at the word-level is still the most effective, and that increasing the depth of such convolutional models with word inputs does not bring significant improvement. Conversely, deep models outperform shallow networks when the input text is encoded as a sequence of characters, but although such deep models approach the performances of word-level networks, they are still worse on the average. Another contribution in this chapter is the proposal of a new deep model that is an adaptation of DenseNet for text inputs.

Based on the litterature and the results presented in this chapter, our main conclusion is that deep models have not yet proven to be more effective than shallow models for text classification tasks. Nevertheless, further researches should be realized to confirm or infirm this observation on other datasets, natural language processing tasks and models. Indeed, this chapter derives from reference deep models that have originally been developed for image processing, but novel deep architectures for text processing might of course challenge this conclusion in the near future.

Chapter 4

Dialogue acts and sentiment analysis

4.1 Introduction

One of the most studied natural language processing task on Twitter is sentiment recognition, which is for example a recurrent task every year at the SEMEVAL evaluation campaign (Nakov et al., 2016b; Bethard et al., 2017). Conversely, there are fewer works that study dialogues on Twitter. Dialogues on social media have quite a different form than on other media, such as vocal, sms and meetings interactions, because the participants in the dialogue are often unknown in advance, they may come and leave at any time and live in different time zones. The structure of social media dialogues thus typically forms a tree, or a directed acyclic graph if we take into account @-mentions. We focus next on dialog acts, also known as speech acts, which characterize the function of a phrase in the course of a dialog. Typical dialog acts are questions, answers, disagreements...

The scientific hypothesis that is studied in this chapter is that sentiments are correlated to dialog acts on social media, and that this correlation may be exploited to enable transfer learning between both tasks. Intuitively, consider a dialog where user A writes something *positive* (sentiment) about a given smart phone brand. Then, user B *disagrees* (dialog act) and points out a *negative* aspect of this brand. An obvious correlation between dialog acts and sentiments can be found on this trivial example. Surprisingly, the majority of works about sentiment analysis ignores such relations between dialog acts and sentiments.

As far as we know, a joint model of sentiments and dialog acts is first proposed in Kim and Kim, (2018). It exploits one convolutional network per task, which output vectors are concatenated and passed to one classifier per task. It makes strong simplifying assumptions to merge the tasks, in particular:

- The dialog act at time t does not depend on sentiments, but does depend on the dialog act at time $t-1$;
- The sentiment at time t only depends on the sentiment and dialog act at time t (no Markov hypothesis);

Despite these strong hypothesis, the authors report better results when considering jointly dialog acts and sentiments on a Korean corpus. However, their corpus and code is not distributed, which makes comparative evaluations with this model difficult. Compared to this work, our proposed network (in Section 4.3) jointly models both tasks at deeper layers, just after the word embeddings layer. Furthermore, our hypothesis are weaker, because both labels at time t depend on both tasks at time t and $t-1$, and recurrence is applied at both word and dialog levels. We propose

next to demonstrate experimentally that this correlation is strong enough to enable transfer learning between both tasks, and we further analyze both quantitatively and qualitatively some of the observed correlation patterns. Finally, we distribute our corpus and software under an open-source and free license to make future comparison easier.

4.2 Mastodon Corpus

Social media are a gold mine for researchers in many domains and especially in natural language processing, because of the endless stream of linguistic content produced every day. However, a major issue faced by every researcher working with Twitter data concerns the accessibility of datasets previously extracted. Indeed, license restrictions limit the possibility to store tweets in a database for a long period of time, and the proportion of tweets that are continuously deleted from the Twitter company servers makes any Twitter corpus quickly obsolete and impossible to retrieve after a few months. This is a very serious issue for the ethics of experimental research, of which reproducibility is a foundational feature. Despite these limitations, many research and development works use Twitter data for a wide variety of applications.

We propose in this work to exploit another social medium with language data, the Mastodon network, which closely resembles Twitter, except for two important differences: Mastodon is a decentralized social network, and it adopts permissive licenses that are helpful with reproducible research. In particular, everyone may create his own Mastodon server with his cohort of registered users, and have his server automatically federated within the worldwide social network. To enable this, the Mastodon software is released with the open-source AGPL license, and, depending on the administrator of the hosting node, the user-generated content in the Mastodon network may in some cases follow a Creative Commons license, which allows redistribution of posts.

Mastodon is very similar to Twitter with regard to content and usage, except that the user posts are limited by default to 500 characters, which gives the user the possibility to write longer sentences than on Twitter. From a sociological point of view, Mastodon users are mostly composed of people who are either attracted by the technical challenge of setting up their own server, or who have been disappointed by Twitter, sometimes because of advertisements, changes of policies and privacy threats, or who feel a strong sense of belonging to a minority group, such as LGBT, because Mastodon naturally enables and favors the emergence of local communities. Another difference, which is relevant for researchers in Linguistics and Natural Language Processing, is that Mastodon is more international than Twitter, in the sense that the majority of Mastodon servers are located in Japan and in the West of Europe (France, Spain and Germany in particular), and although these servers are federated together, each of them brings together a community, which is often linguistic. It is thus frequent to observe on some server a majority of posts that are not written in English.

In terms of the amount of language data produced every day, Mastodon is still less developed than Twitter, but it is growing and is already large enough (with one and a half million of users as of February 2018) to be useful for most research works. We demonstrate this by scrapping and annotating a corpus of dialogues in English from Mastodon and training and validating a deep learning model on this data.

Both the (manually anonymized) corpus and software described in this work are distributed freely at `https://github.com/cerisara/DialogSentimentMastodon`.

We have crawled about 800,000 posts from the *octodon.social* Mastodon instance, which is one amongst thousands of existing instances, and have filtered out non English posts automatically with the *langdetect* python library. We have then followed the *reply-to* links to structure all posts into dialog trees: dialogues in social media are structured as trees, because anyone can get involved in a dialog from any post that composes the dialog so far. About half of the dialogues have two posts, 1/4 three, and so on. The longest dialog is composed of 44 posts. The tree-dialogues are then split into a training and test set.

Before annotation, all dialogues are linearized, following the work of Zarisheva and Scheffler, (2015), which means that each branch of the tree forms a unique dialog. Two students with a Master degree in linguistics and fluent in English independently assigned two tags to each post in a dialog:

- A sentiment tag with 3 possible values: positive (26% of the corpus), negative (31%) and neutral (43%); The baseline performances, when always classifying posts as neutral, are F1=24.4%.
- A dialog act with 27 possible values (the bold labels on the right of Table 4.2).

A typical dialog is shown in table 4.1.

	Sent	**DA**	**Textual content of segment**
0	-	I	because this is getting way too much attention it wasn't even that funny URL
1	+	I	LMAO
2	*	O	why you still on twitter though ?
3	-	W	i'm trying to get all my other friends to get on here before i deactivate !
4	-	I	I might have to do that too . Some ain't migrated yet . It's an issue .
5	-	I	some of my mutuals are waiting for more leftists go get on here
6	*	O	and i'm like what are you waiting for ? ? ?
7	-	I	then again i did have to spend hours to figure out how to work this site haha
8	-	Q	lol was it that hard ? ?
9	-	W	i'm not that good with technology or websites
10	-	A	so yeah ..
11	-	A	well yeah , the whole instances thing is kind of confusing tbh .

TABLE 4.1: Example of a typical dialog from the Mastodon corpus.

The dialog act tags have been derived from the seminal work on the Switchboard corpus (Core and Allen, 1997) and have been adapted to take into account specificities of social media, taking inspiration from the work of Zarisheva and Scheffler, (2015). The recently proposed ISO standard (Bunt et al., 2017) has also been taken into account in the design phase of the annotation guide. This process has lead to the definition of the tags listed in Table 4.2. After a first round of annotations and in order to avoid rare labels, these 27 tags have been further merged into 15 tags, whose distribution is shown in Table 4.2.

The inter-annotator agreement is 88.6% for dialog acts and 90.2% for sentiments. The Cohen's kappa coefficient is 85.1% for dialog acts and 90.2% for sentiments. The

final training corpus is composed of 239 dialogues for a total of 1075 posts, and the test corpus of 266 dialogues for a total of 1142 posts. The vocabulary size is 5330 words.

4.3 Multi-task model

4.3.1 Model description

Our proposed model is shown in Figure 4.1. It is a two-level hierarchical recurrent network similar to the one proposed in Sordoni et al., (2015):

- The first level (*post level*) takes as input the sequence of word embeddings in a post. This first level is composed of a bi-LSTM, which outputs a single vector per post.
- The second level (*dialog level*) takes as input the sequence of vectors produced at the first level, i.e., one vector per post for every post within a dialog. The second level is composed of a standard RNN, because the length of dialogues rarely exceeds 10 posts and so the LSTM cells do not offer a clear advantage over standard recurrent cells. The output of this RNN at every post is passed to two MLPs, one for dialog acts and another for sentiment labels.

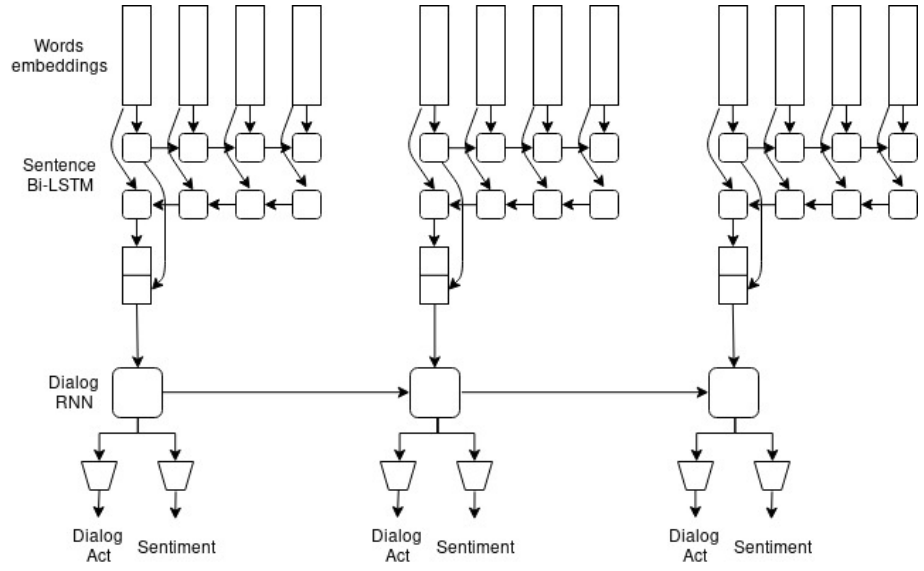


FIGURE 4.1: Multi-task hierarchical recurrent model. The number of posts per dialog and the number of words per post may vary and are handled dynamically without any padding.

4.3.2 Training procedure

Two cross-entropy losses are used to train the model, one for dialog act recognition and another for sentiment classification. In our corpus, every post is manually annotated with both a sentiment and dialog act labels. So standard multi-task training simply consists in backpropagating the gradient of both losses with equal weights. However, we also realize some transfer learning experiments with fewer annotations in one of the tasks. In such cases, we artificially remove the gold sentiment label for

training and only backpropagate the gradient of the dialog act loss, and vice versa when transferring from the sentiment to the dialog act task.

A development corpus is extracted using 10-fold cross-validation. Three values of the learning rate (0.1, 0.01 and 0.001) have been tried on this development corpus and the best one has been kept. The number of epochs, up to a maximum of 500 epochs, is also tuned on this development corpus for each experiment. Furthermore, because some badly initialized weights may fail to converge, every experiment on the development corpus is run twice with different random initializations, and the best one on the development set is kept. Running only twice does not guarantee that at least one training is correct, but it reduces the chances of failure, without increasing too much the computation requirements. All other hyper-parameters have been set a priori to reasonable values: 100-dim LSTM hidden state size, 100-dim word embeddings, 0.4 dropout, ReLU activations.

4.4 Evaluation

Model evaluation is performed with metrics used in related works:

- For sentiment recognition, following SemEval 2016 (Nakov et al., 2016b), this is the macro-average of the positive and negative F1 scores;
- For dialog act recognition, following Zarisheva and Scheffler, (2015), this is the average of the dialog-act specific F1 scores weighted by the prevalence of each dialog act.

The model has been written in pytorch. The source code, along with the manually anonymized data used in this work, is released as open-source and is available at `https://github.com/cerisara/DialogSentimentMastodon`.

4.4.1 Multi-task experiments

The development corpus is composed of 28 dialogues (average size across all 10 folds). We experiment next with a training corpus of varying size: 1, 10, 50, 100, 150, 200 and the full set of 239 dialogues. The x-axis in Figures 4.2 and 4.3 are labeled with the total number of annotated dialogues used at training time (train + dev set). The number of epochs for training is tuned on the development set by optimizing either the sentiment loss (**target sentiment**) or the dialog act loss (**target dialog act**).

In Figure 4.2, both the mono-task and multi-task models have similar performance. So considering an additional label for the second task does not help, as compared to when only the label of the target task is given. This might be due to the fact that the model already captures all available information with a single task, and the auxiliary label does not bring enough new information to help recognition of the target label.

4.4.2 Transfer between tasks

We study next unbalanced cases, when the number of annotations differ between tasks. Figure 4.3 compares the evolution of the sentiment analysis F1 score when:

- Both tasks are trained on the same number of annotated dialogues (**both-rich**); this is the same experiment as realized in Section 4.4.1.

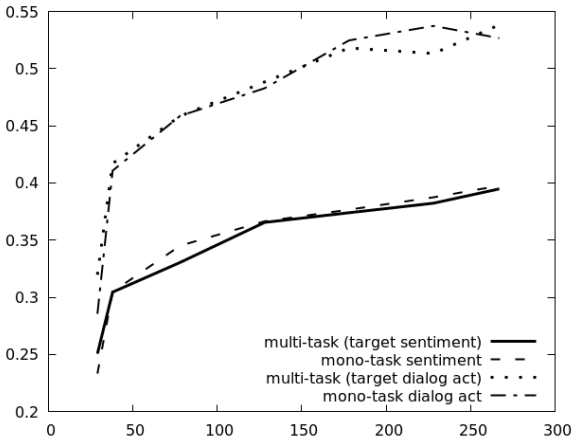


FIGURE 4.2: Sentiment and dialog act F1s as a function of the number of dialogues used for training.

- The sentiment recognition task is limited to a maximum of 38 (10 dialogues for training plus the development corpus) annotated dialogues, while the training size of the dialog act task is not limited (**sentiment-poor**);
- Both tasks are limited to only 38 annotated dialogues (**both-poor**).

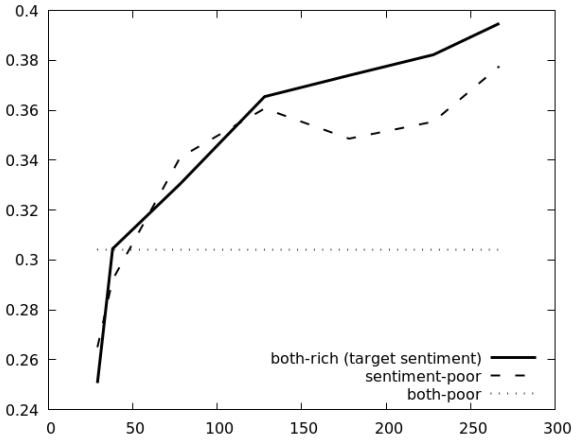


FIGURE 4.3: Sentiment F1 as a function of the number of dialogues used for training. **both-rich:** both tasks have the same training size; **sentiment-poor:** only 38 dialogues maximum are annotated with sentiment labels; **both-poor:** both tasks are limited to 38 training dialogues.

The curves show that information is largely transferred from the richer dialog act recognition task to the target sentiment classification task. Hence, although the **sentiment-poor** and **both-poor** systems have only access to 38 dialogues annotated with sentiment labels, the accuracy of the **sentiment-poor** model keeps on increasing when additional dialog act labels are considered. On the right, when the full dialog act training corpus is used, its accuracy is quite close to the one obtained with all sentiment labels, and is better than the **both-poor** model by a large margin, thanks to multi-task transfer learning.

Conversely, we also validate transfer learning from a rich sentiment recognition task to a poor dialog act recognition task. The resulting curves are shown in Figure 4.4.

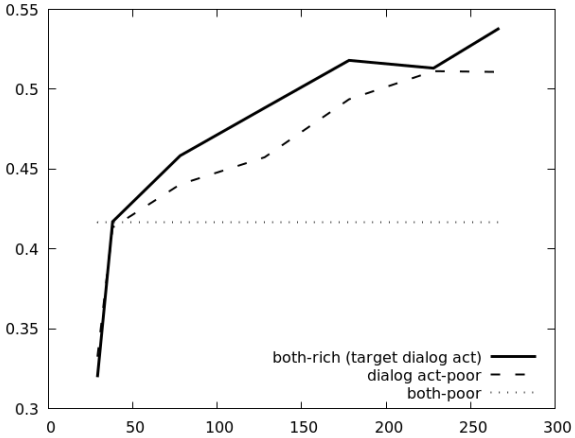


FIGURE 4.4: Dialog act F1 as a function of the number of dialogues used for training. **both-rich:** both tasks have the same training size; **dialog act-poor:** only 38 dialogues maximum are annotated with dialog act labels; **both-poor:** both tasks are limited to 38 training dialogues.

Similar gains in performance are obtained through transfer learning for the dialog act recognition task than for the sentiment analysis task. We can thus conclude that transfer learning is efficient between both tasks in both directions.

4.5 Analysis

Beyond quantitative experiments that demonstrate transfer learning between dialog act and sentiment recognition, we analyze next some properties of our Mastodon corpus.

Dynamics of sentiments and dialog acts

In the course of a dialog, the sentiment globally changes at a slower rate than the dialog act. While dialog acts change nearly at every segment, often, a single sentiment is expressed per dialog, sometimes two or three, but rarely more. This is understandable, but this also implies that the correlation between both tasks is not very strong, otherwise, they would have a much similar dynamic. On the other hand, they are not completely independent, as shown in the transfer learning experiments. So we try and exhibit next some patterns where correlation between tasks is explicit in the corpus.

Both tasks are sparsely correlated

In order to validate our initial hypothesis that, in the course of a dialog, sentiments may be correlated to agreements and disagreements, we have computed the transition log-probabilities between the previous sentiment s_{t-1} and the current sentiment s_t as a function of the current dialog act d_t. These log-probabilities are shown respectively for $s_{t-1} =$ neutral, positive and negative in Figures 4.5 and 4.6.

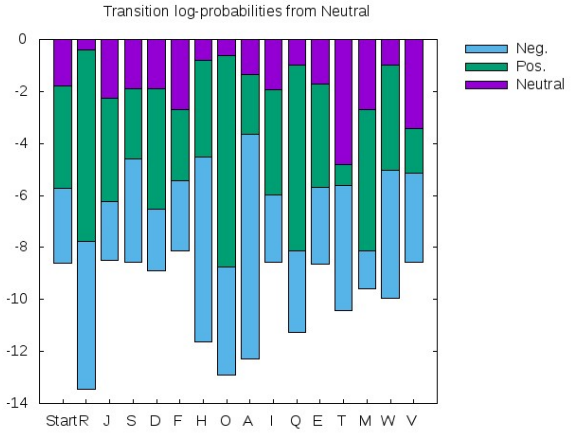


FIGURE 4.5: $\log p(s_t|d_t, s_{t-1} = neutral)$. The dialog acts d_t are listed at the bottom of each histogram column. Because log-probabilities are plotted, boxes with the smallest height are the most likely. The start of a dialog, which has no previous sentiment, is a special case that is added on the left.

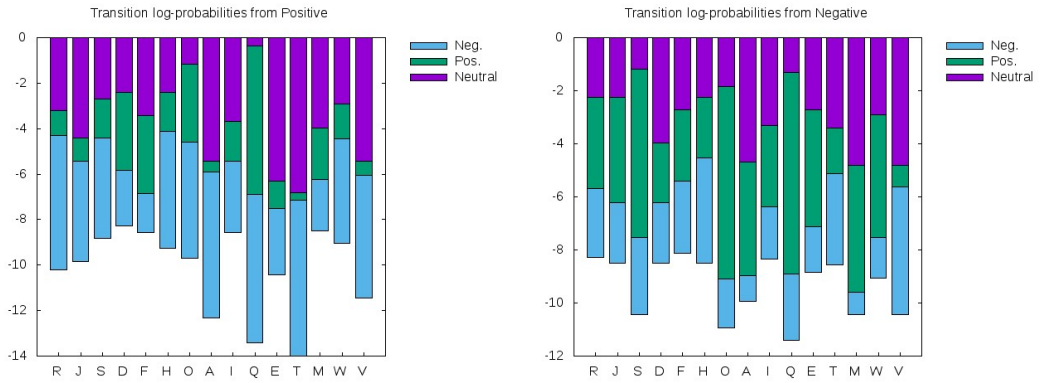


FIGURE 4.6: $\log p(s_t|d_t, s_{t-1} = positive)$ (left) and $\log p(s_t|d_t, s_{t-1} = negative)$ (right)

We observe that:

- With agreements (**A**), a positive sentiment mainly stays positive and a negative sentiment mainly stays negative.
- With disagreements (**D**), a positive sentiment becomes either neutral or negative, a negative sentiment becomes either positive or stays negative, a neutral sentiment either stays neutral or becomes negative.

These observations globally follow our intuition, even though more complex patterns than expected seem to occur. Another observation concerns the evolution of sentiments from the start to the end of a dialog. Previous studies have shown that, with regard to dialogues, the sentiment on social media is related to:

- The topic of discussion: for instance, discussions about politics tend to have more negative sentiments (Thelwall, Buckley, and Paltoglou, 2012);

- The length of the dialog: negative sentiments tend to lead to longer dialogues (Thelwall, Sud, and Vis, 2012), which we suggest may be also related to the correlation between sentiments and disagreements that we have observed in this work; indeed, an increased proportion of disagreements may lead to longer dialogues.

Based on our corpus, another pattern seems to emerge, which is in favor of negative sentiments at the beginning of dialogues and positive sentiments at the end. This is intuitively plausible, as at least a small proportion of the sources of discords are likely to be resolved during dialogues, but this interpretation is still to be confirmed on a more diverse set of corpora.

4.6 Conclusion

We have studied a multi-task model for joint sentiment and dialog act recognition on social media. We have shown that transfer learning is quite efficient when the number of annotated labels for one task is smaller than for the other task. We have further analyzed the correlation between both tasks and shown that although there is enough mutual information to enable transfer learning, both tasks are not strongly correlated globally. They are actually characterized by different dynamics, but we have nevertheless found a few specific patterns that exhibit a strong correlation. All these studies are realized on a new corpus extracted from the Mastodon social network. Both the annotated corpus and source code of our model are available with an open-source license on github.

In this work, I have mainly contributed by co-supervising the two linguists Master students who have defined the set of dialog acts and the annotation guide, and contributing to the brainstorming sessions about the state-of-the-art and design of the model.

- Communicative functions
  - General purpose functions
    - Information seeking functions
      - Question
        - Propositional question
          - Yes/No question ..... **Q** (8.3%)
          - Check question ..... (merged a priori in Q)
        - Set question
          - Wh* / Open question ..... **O** (7.4%)
        - Choice question
          - Open with choices ..... (merged a priori in O)
    - Information providing functions
      - Inform
        - Statement ..... **I** (49.3%)
        - Agreement ..... **A** (7.9%)
        - Disagreement ..... **D** (1.9%)
          - Correction ..... (merged a priori in D)
        - Answer
          - Open + choice answer ..... **W** (9.9%)
          - Confirm answer ..... **Y** (merged in A)
          - Disconfirm answer ..... **N** (merged in D)
    - Commissive functions
      - Offer ..... **E** (1.4%)
        - Promise ..... (merged a priori in E)
    - Directive functions
      - Request ..... **R** (3.3%)
        - Instruct ..... (merged a priori in R)
      - Suggest ..... **S** (3.0%)
    - Directive & commissive functions
      - Accept offer request suggest ..... **P** (merged in A)
      - Decline offer request suggest ..... **L** (merged in D)
  - Feedback functions
    - auto-positive acknowledgement ..... **F** (0.2%)
    - allo-positive acknowledgement ..... (merged a priori in F)
    - auto-negative acknowledgement ..... **B** (merged in F)
    - allo-negative acknowledgement ..... (merged a priori in B)
  - Social obligation functions
    - Initial greetings ..... **H** (2.0%)
    - Return greetings ..... (merged a priori in H)
    - Initial goodbye ..... **G** (merged in H)
    - Return goodbye ..... (merged a priori in G)
    - Apology ..... **X** (merged in M)
    - Accept apology ..... **C** (merged in A)
    - Thanking ..... **T** (2.0%)
    - Accept thanking ..... **K** (merged in A)
- Exclamation ..... **J** (1.5%)
- Explicit performative (*hope you get better*) ..... **V** (1.6%)
- Sympathy (*I'm sorry to hear that*) ..... **M** (0.6%)
- Miscellaneous / other ..... * (absent)
- UNK (too ambiguous to decide) ..... **U** (removed)
- Malformed input ..... **Z** (removed)

TABLE 4.2: List of dialog acts with the 27 labels used by annotators (letters in bold) and their distribution in the corpus (percentage within parentheses). Some labels have been merged a priori (before the annotation process), in order to simplify the annotation process. Very rare labels have been merged after the annotation process. The remaining 15 labels used to compute the F1 metric are the ones with a distribution percentage. The baseline recognition performance obtained when always answering **I** is F1=35%.

Part II. Sentence Compression

Chapter 5

Related work

5.1 Neural Sentence Summarization

Sequence-to-sequence (s2s) is a popular model used for summarization task but is still far from perfect. First of all, it largely shares common problems of neural machine translation (Koehn and Knowles, 2017). Further, text summarization contains its own difficulties that s2s model can not easily cope with. For instance, Amplayo, Lim, and Hwang, (2018), Cao et al., (2017), Song, Zhao, and Liu, (2018), Li et al., (2018a), and Li et al., (2018b) observed that sequence models can produce incorrect, hallucinated and non-factual output. A common remedy often used consists to integrate additional structural bias to make sure that the attention of the model spreads over key information in the source.

Amplayo, Lim, and Hwang, (2018) observed that information around entities are related to the topic of the summary. When producing summaries, it is crucial to concentrate around the main topics and omit unnecessary information while s2s model tends to consider all the information in the original text, regardless of being appropriate or not. The problem is more pronounced when summarizing longer texts as the attention mechanism may focus on irrelevant topics. Moreover, understanding properly these entities requires extra commonsense information as context which are often not included in the source text. Thus, Amplayo, Lim, and Hwang, (2018) proposed the Entity2Topic module, which associates linked entities from a knowledge base into the summarizer as a topic module to guide the decoding process.

In another study, Cao et al., (2017) also pointed out that nearly 30% of the generated summaries from a state-of-the-art s2s system often mismatch with the original texts and produce fake facts. An illustrative example of the output from the s2s model (Nallapati et al., 2016) is shown in Figure 5.1. While "repatriation" is the true subject of "postponed", the system picked "bosnian moslems" as the subject, probably because it is closer to "postponed" in the source, and falsified a fact "bosnian moslems postponed". Also, the system added another fake fact: "unhcr pulled out of bosnia" into the summary. As a consequence, whereas the readability and informativeness (Rouge-1 F1=0.57) of this output are high, the faithfulness to the original text is not respected. This hallucination of summaries is a well-known problem of all s2s models.

To alleviate this issue, Cao et al., (2017) proposed to use open information extraction and dependency parsing technique to infer actual facts from the source text and force the generator to respect these descriptions. Specifically, they first extracted the facts from the source sentence which is represented either by a relation triple (subject; predicate; object) computed with the Open Information Extraction (OpenIE) system (Banko et al., 2007), or from the dependency parse tree with the (subject; predicate) and (predicate; object) tuples. A sentence and its dependency parsing tree are shown in Figure 5.2 as an example. While OpenIE yields empty

Source	the repatriation of at least #,### bosnian moslems was postponed friday after the unhcr pulled out of the first joint scheme to return refugees to their homes in northwest bosnia .
Target	**repatriation** of bosnian moslems postponed
s2s	**bosnian moslems** postponed after unhcr pulled out of **bosnia**

FIGURE 5.1: An example of fake summaries generated by the state-of-the-art s2s model. *Credit: (Nallapati et al., 2016).*

output, the dependency parser will generate the following predicate-related tuples: (prices; opened) (opened; tuesday) (dealers; said) and the modify-head tuples: (taiwan; price) (share; price) (lower; tuesday). From the tuples, two fact descriptions will then be merged via the symbol | to form a supplement input source text to the model: *taiwan share prices opened lower tuesday ||| dealers said*.

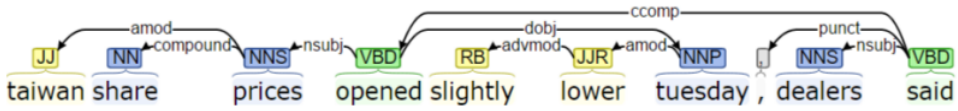


FIGURE 5.2: A dependency tree example. Cao et al., (2017) extracted the following two fact descriptions: *taiwan share prices opened lower tuesday ||| dealers said*. *Credit: (Cao et al., 2017).*

Then, the authors use dual-encoders to read both the sentence and fact descriptions simultaneously, compute their respective attention vectors, integrate them proportionally into a common context vector and finally generate the summary. The whole process of the proposed FTSum model is shown in Figure 5.3.

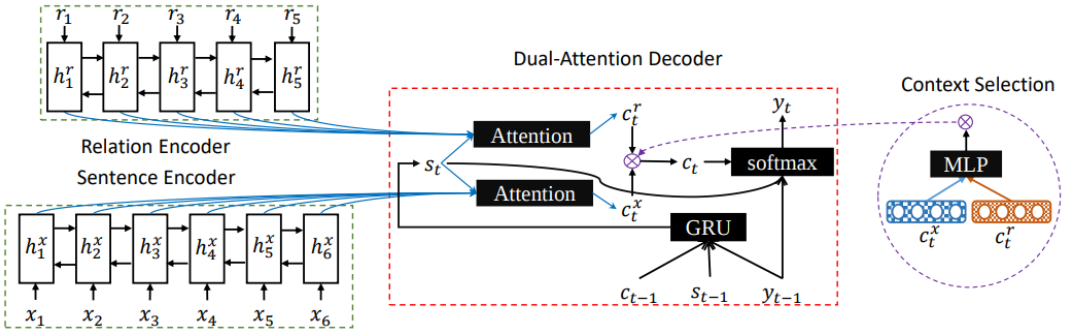


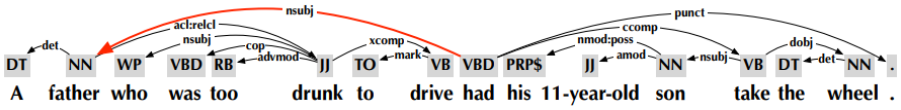
FIGURE 5.3: FTSum model architecture. *Credit: (Cao et al., 2017).*

In another work, Song, Zhao, and Liu, (2018) found that the s2s model fails to preserve the meaning of the original texts also partly because it is not aware of the syntactic structure of the source sentence. Figure 5.4 presents an example. This sentence has an appositional clause *"who was too drunk to drive"* located between the subject and the main verb. However, the s2s model does not recognize the main verb and picks instead the first few words of the source texts. In this case, as the syntactic structure of the source sentence is relatively rare, Cao et al., (2017) predict that it may pose issues for neural summarizers, which may not be able to learn such rare syntactic patterns. Furthermore, the s2s model anlyzes the input sequentially and is not explicitly aware of the syntactic structure of the source sentences.

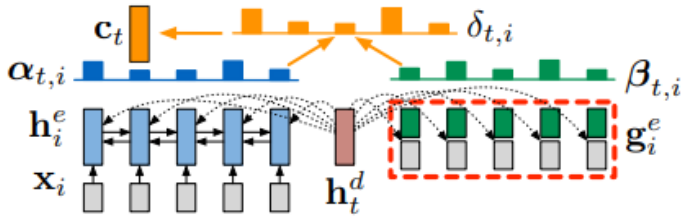
Src	An Alaska father who was too drunk to drive ***had*** his 11-year-old son take the wheel, authorities said.
Ref	Drunk Alaska dad ***has*** 11 year old drive home
Sys	alaska father who was too drunk to drive

FIGURE 5.4: The sequence-to-sequence model fails to preserve summary-worthy content of the source (e.g., main verbs) despite their syntactic importance. *Credit: (Song, Zhao, and Liu, 2018).*

Thus, they sought to tackle this issue by combining the source syntactic structure into the neural summarizer to allow the system to identify summary-worthy content. More specifically, they injected structural information into the copy mechanisms to give more preference towards copying source words and relations from source to the summary according to their semantic and structural importance in the source texts. Figure 5.5b shows a strategy used in this study. To model the structural importance of source texts, they calculated another attention distribution on the right side of Figure 5.5b by comparing each structure-enhanced embedding g_i^e with the decoder hidden state h_t^d, in addition to the common attention of semantic distribution on raw texts on the left. With a further small tweak of adding dependency edge among words as embeddings to this architecture design, they can show that the main verb and the subject ("father" ← "had" in Figure 5.5a) can be conserved in the summary.



(A) An example dependency parse tree. If important dependency edges such as "*father* ← *had*" can be preserved in the summary, the system summary is likely to preserve the meaning of the original.



(B) System architectures of 'Struct+2Way+Word'. *Credit: (Song, Zhao, and Liu, 2018).*

FIGURE 5.5

Following this paradigm of keeping key information, Li et al., (2018a) employed the TextRank unsupervised algorithm (Mihalcea and Tarau, 2004) to extract keywords from the source text and used them to guide the generation process. With a similar goal, Li et al., (2018b) took another radical approach of entailment relations to ensure the correctness of the summary. Arguing that a summary is only correct if it is semantically entailed by the source sentence, they proposed to incorporate entailment knowledge into the neural summarizer. First, they shared the s2s encoder with an entailment recognition task on an entailment corpus to improve the correctness aspect of the summarizer. Then, on the decoder side, they used Reward Augmented Maximum Likelihood (RAML) (Norouzi et al., 2016) training to further encourage the system to produce summary entailed by the source. Though this attempt can show promising result of improving the faithfulness of the summarization model, it nevertheless makes the system more obscure and harder to interpret than

the syntactic approach.

In parallel, another research path aims at directly learning to distill out important sentences from the source document making the neural sequence model more interpretable. This is applied specifically on the CNN/Daily Mail dataset. Nallapati, Zhai, and Zhou, (2017) proposed SummaRuNNer, a model for extractive document summarization that allows a very intuitive visualization of the prediction and decision of its abstract features (i.e., information content, salience and novelty). Figure 5.6 shows the architecture of the model.

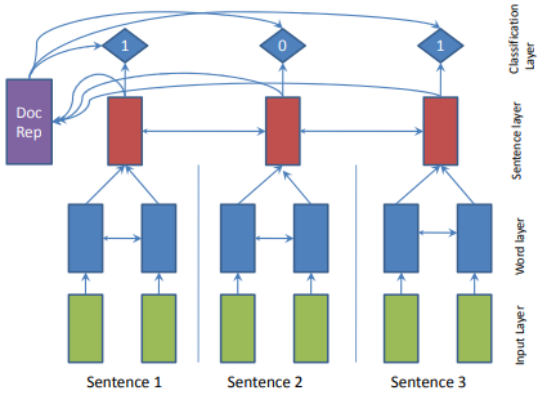


FIGURE 5.6: SummaRuNNer extractive summarization model. *Credit: (Nallapati, Zhai, and Zhou, 2017).*

In a nutshell, they casted extractive summarization as a sequence classification problem: each sentence in the source document is visited in a sequential order and a decision of keeping or excluding from the summary is made. Specifically, the model consists of two layers of bi-RNNs on word and sentence level. Average-pooling of hidden states on words forms the input of the next layer of the network, where it encodes the representations of the sentences in the source document. Then, each sentence is visited in order again by a logistic layer to decide if that sentence should be kept in the summary. The choice of each sentence relies mainly on the sentence's content richness, salience with regard to the document, and its novelty with regard to the accumulated summary representation.

Though being more interpretable, the previous model suffers two issues. First, they only picked greedily the sentences with the highest sorted predicted probabilities until exceeding the length limit *at test time* and did not learn these decisions at training time. Secondly, there's a discrepancy between the cross-entropy learning objective and the evaluation criterion. Thus, Narayan, Cohen, and Lapata, (2018) proposed to cast extraction as a ranking problem and used reinforcement learning to optimize directly the final objective. Their model (Figure 5.7) first composes of a hierarchical document encoder (a refinement of sentence encoder) and a hierarchical sentence extractor playing the roles of encoding sentence representation and sequentially labeling them (keep or remove) as before. Moreover, they ranked each sentence in the document by the score given by the softmax layer of the sentence extractor. The ranking system is trained by a reinforcement learning framework, directly optimizing the final evaluation ROUGE metric (Lin, 2004). Consequently, the model distinguishes better sentences as a sentence is ranked high only if it often occurs in high scoring summaries.

Though alleviating many issues of common neural extractive summarizers, the generated output of the aforementioned model is still purely extractive and not elegantly human-readable. On another spectrum, abstractive models mimick better

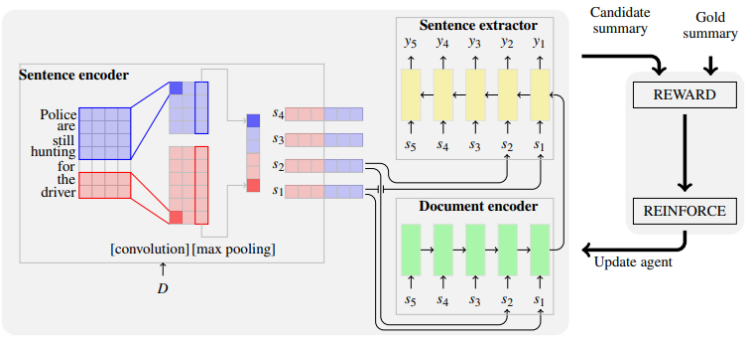


FIGURE 5.7: Reinforcement Learning-based Extractive Summarization. *Credit: (Narayan, Cohen, and Lapata, 2018).*

the way that humans summarize long documents, which first select salient information and then paraphrase them abstractively. However, common neural abstractive summarizers (See, Liu, and Manning, 2017; Paulus, Xiong, and Socher, 2018) tend to be slow when encoding very long documents and inaccurate of determining important phrases from a long chain of words via attention mechanism. To tackle these problems and combine the best of both worlds, Chen and Bansal, (2018) proposed a hybrid extractive-abstractive model (Figure 5.8), arguably better approximation of human behavior. In short, the model takes extractive method as an intermediate step to improve the overall model's speed, quality and uses a decoder to rewrite abstractively and generate new words from the full vocabulary.

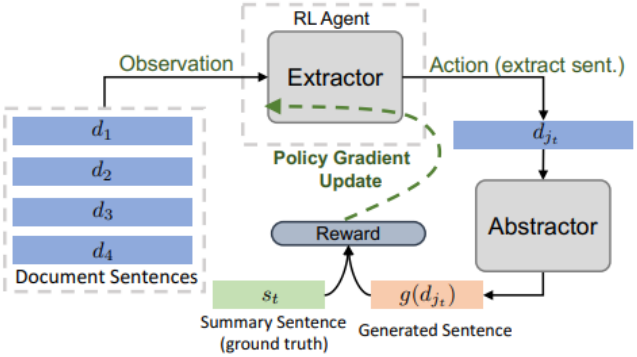


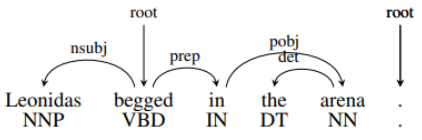
FIGURE 5.8: Hybrid extractive-abstractive model. *Credit: (Chen and Bansal, 2018).*

Concretely, the model first adopts the coarse-to-fine extraction approach that is described before, which chooses all the salient sentences and then rewrites each of them parallelly. Finally, as this *hard* extraction is a non-differentiable behavior, a policy gradient algorithm with ROUGE metric as rewards is applied to train the whole model end-to-end. Intuitively, the extractor agent will encourage (or discourage) the action of choosing a sentence if the abstractor produces a good (or bad) match with the ground truth. This architecture paradigm is also good at saving the summarizer from common redundancy issues of neural abstractive models as only neat and non-unnecessary salient sentences are already chosen to summarize.

5.2 Neural Text-to-text Generation

As there are numerous interesting topics on text-to-text generation, we will only discuss syntax-aware and reinforcement learning (RL)-based models, which are common techniques used for summarization (and machine translation). Understanding well the evolution of these two approaches is essential for the study of one to empirically explain the other, that we will conduct in the subsequent part of the thesis.

Syntax models: While a sequence-to-sequence model may achieve remarkable results of learning on aligned parallel raw text, it does not tend to be accurate of small variations of word forms. Inspired by Collobert and Weston, (2008) on classification tasks, explicit modeling of syntax has frequently been used in text generation applications such as machine translation and summarization. Sennrich and Haddow, (2016) enriched the input to machine translation with dependency labels, POS tags, subword tags and lemmas so that each input token is represented by the concatenation of word and features embeddings (Figure 5.9). Similarly, Nallapati et al., (2016) enriched the encoder side of a neural summarization model with POS tag, NER tag, TF-IDF features. The intuition is that words will be better disambiguated by taking syntactic context into account.



words	Le:	oni:	das	beg:	ged	in	the	arena	.
lemmas	Leonidas	Leonidas	Leonidas	beg	beg	in	the	arena	.
subword tags	B	I	E	B	E	O	O	O	O
POS	NNP	NNP	NNP	VBD	VBD	IN	DT	NN	.
dep	nsubj	nsubj	nsubj	root	root	prep	det	pobj	root

FIGURE 5.9: Original dependency tree for sentence *Leonidas begged in the arena .*, and feature representation after BPE segmentation. *Credit: (Sennrich and Haddow, 2016).*

Speculating that full parse trees can be more beneficial than shallow syntactic information, Li et al., (2017) integrated to the input a linearization of its parse tree. Figure 5.10 shows two ways for this integration: Parallel RNN and Hierarchical RNN Encoder. First, a parse tree is linearized in a depth-first traversal order into a sequence of structural labels. Then, Parallel RNN Encoder includes two RNNs of word and structural label in a parallel way such that the backpropagation error from one will not affect the other. Hierarchical RNN Encoder puts the structural label layer under the word layer so that the word embedding is modified directly from the structural label embedding.

Other studies have focused on integrating syntax in the decoder through multi-task learning. Le et al., (2017) defined machine translation as a sequence-to-dependency task in which the decoder generates both words and a linearized dependency tree while Kiperwasser and Ballesteros, (2018) proposed a scheduled multi-task learning approach where the main task is translation but the model is alternatively trained on POS tag, Dependency Tree and translation sequences.

RL sequence models: Early neural text generation models use *teacher forcing* with cross-entropy loss as common training technique to learn features automatically. This paradigm suffers from two major drawbacks. First, during the training time, the generation of the next word is based on the previous correct token. However, at

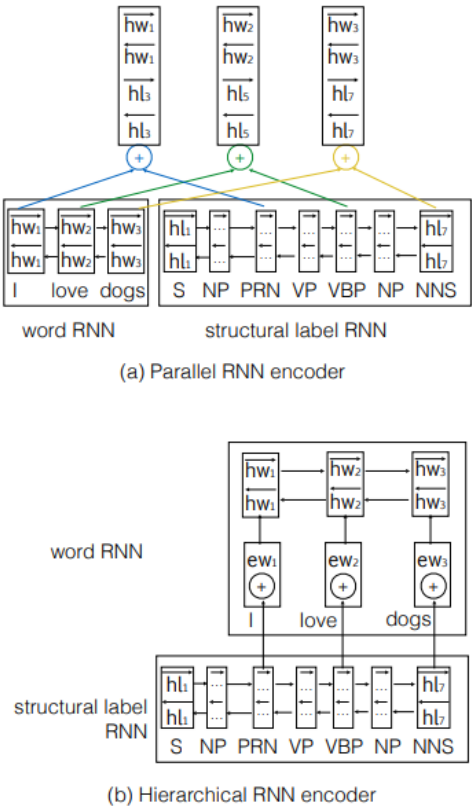


FIGURE 5.10: Modeling source syntax. *Credit: (Li et al., 2017).*

test time, as lacking of the ground-truth, the model is conditioned on its own predictions. As a result, this discrepancy between training and testing paradigm make errors accumulate along the way, especially for long sequences. This suboptimal behavior of training the model on data distribution instead of its own predictions is regarded as *exposure bias* problem (Figure 5.11).

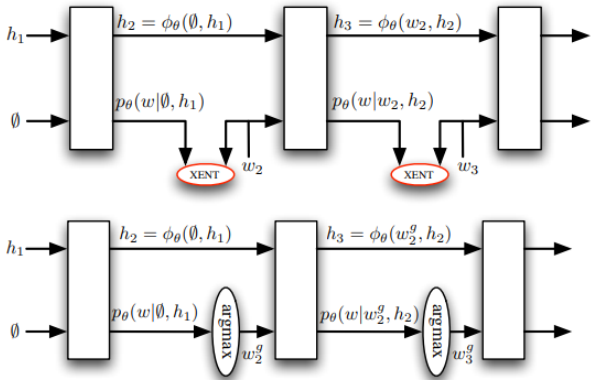


FIGURE 5.11: RNN training using cross-entropy (top), and how it is used at test time for generation (bottom). *Credit: (Ranzato et al., 2016).*

Second, cross-entropy loss is popularly employed as an objective function to maximize the chance of the next correct token. However, evaluations of these models are typically performed by discrete and non-differentiable metrics of n-grams such as: BLEU for machine translation (Wu et al., 2016), ROUGE for text summarization systems (Paulus, Xiong, and Socher, 2018; Narayan, Cohen, and Lapata, 2018) and

SARI (Xu et al., 2016) for sentence simplification models (Zhang and Lapata, 2017). This is known as the *loss mismatch* problem.

In their seminal work, Daumé, Langford, and Marcu, (2009) proposed that the REINFORCE algorithm (Williams, 1992) is a good fit to these problems. The REINFORCE algorithm naturally samples and lets the model use its own predictions at training time. Moreover, it enables direct optimization of any evaluation metric. As a first attempt, they proposed an approach which includes a meta-algorithm but it is generally intractable to compute this oracle for text generation, unfortunately. This issue was first tackled by a simple sequence-to-sequence based algorithm called Data As Demonstrator (DAD) which mixes the ground truth training samples with model predictions (Venkatraman, Hebert, and Bagnell, 2015; Bengio et al., 2015). Figure 5.12 illustrates the idea. At each decoding step, DAD takes with a certain probability either the ground truth data or the prediction from the model at the previous step as input.

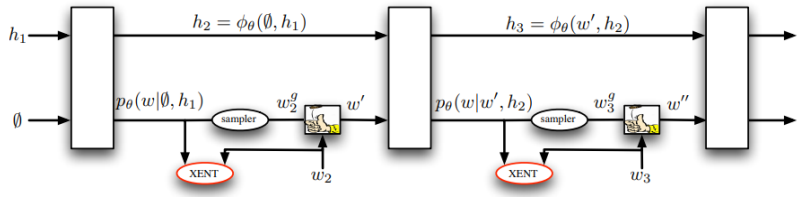
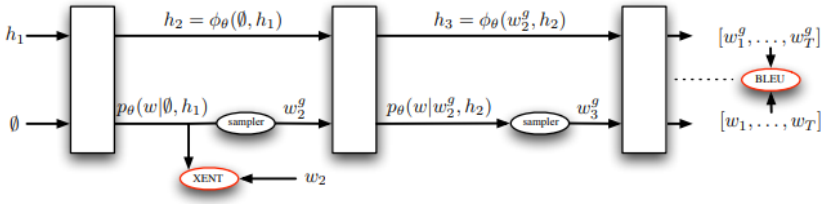


FIGURE 5.12: RNNs trained with cross-entropy and Data as Demonstrator (DAD). *Credit: (Ranzato et al., 2016).*

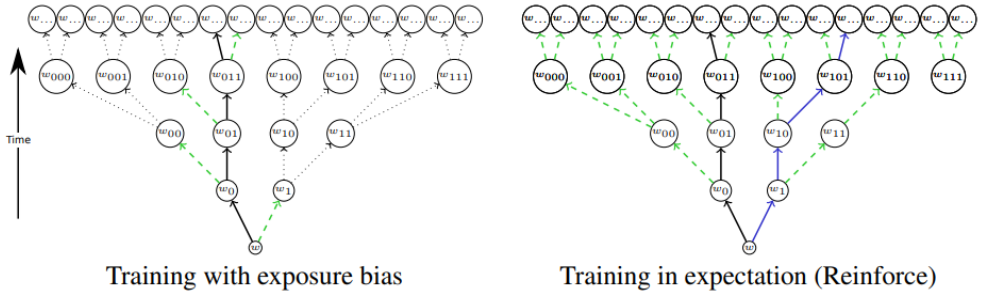
Bengio et al., (2015) later proposed multiple annealing schedules to choose the ground truth words. At the beginning, the model uses the ground truth words and gradually the model predictions are chosen more often as the training progresses. This forces the model to explore more and gradually deal with its own errors during training, as it must do during test time. Despite of being very simple to implement, a major drawback of DAD is that the gradients are not backpropagated through the model sampling. Conditioned on whatever the input, the ground truth data is always selected at each time step. As a consequence, the model is forced to predict a potentially faulty sequence when the targets are not aligned with the generated sequence. For example, if the ground truth sequence is *"Let's go to the cinema"* and the model has predicted *"Let's go to cinema"*, DAD will make the model to predict the word *"cinema"* again. Last but not least, the cross-entropy loss in the algorithm is still a word-level optimization metric.

In a pionner work, Ranzato et al., (2016) introduce an adaptation of REINFORCE (Williams, 1992), called MIXER, that can deal well with the sequence generation problem and alleviate the technical issues of Daumé, Langford, and Marcu, (2009) as follows. The RNN decoder is considered as an agent, which interacts with words and context vector at each time step. The policy is defined by the agent's parameters. After predicting the next word and taking this action, the agent updates its RNN hidden states. A reward is computed (i.e. BLEU, ROUGE) once the agent arrives at the end of a sequence. REINFORCE then minimizes this negative expected reward and backpropagates the gradients through the sampling sequence. Figure 5.13a illustrates the algorithm. Let's see an example of binary vocabulary and all possible sequences of length 4 in Figure 5.13b. The ground truth sequence is the solid black line. Figure on the left presents greedy training approach where the model considers only the probability of the next word. It experiences exposure bias problem as it envisages only surrounding words in the branch marked by green arrows. In the

meanwhile, MIXER on the right figure optimizes over all possible sequences. It uses the predictions made by the model itself during training as it does at test time. The result from the model is a single sample path marked by the blue solid line. Though REINFORCE is a good fit to text generation problem, it suffers from a very large action space. Specifically, as the example in Figure 5.13b shows, the search space is $O(WT)$ where W is the vocabulary size (normally 10^4 or more) and T is the sequence length (around 10-30). To assure the stability and the convergence of the training algorithm, MIXER does not start training from scratch but goes from a trained model with optimal policy given by cross-entropy loss and gradually presents more and more the model to its own predictions.



(A) Mixed Incremental Cross-entropy Reinforce (MIXER). *Credit: (Ranzato et al., 2016).*



(B) Search space for the toy case of a binary vocabulary and sequences of length 4. *Credit: (Ranzato et al., 2016).*

FIGURE 5.13

As the REINFORCE gradient estimator is known to yield high variance, Bahdanau et al., (2016) proposed another reinforcement learning model, namely actor-critic, to lower this variance. Besides the natural decoder *actor*, it trains another *critic* network to estimate the value of each word generated by the policy of actor network. The estimation of this algorithm is sharper but offset by a little bias. The impact of the bias, as well as the goodness of the model, relies particularly on the careful design of this *critic*. In practice, to ensure convergence, intricate techniques must be used including an additional target network Q', delayed actor, a critic value penalizing term and reward shaping.

In contrast, Rennie et al., (2017) introduced a very simple and effective way to reduce this variance for REINFORCE, namely self-critical sequence training (SCST) method. Instead of constructing a baseline for vanilla REINFORCE or using a real critic as above, SCST uses its own prediction normally used at inference time to construct the sequence and uses this to normalize the reward (Figure 5.14). As a result, samples from the model that perform better than the current evaluation-time system are given higher reward and will be pushed up in terms of probability while those samples with inferior reward will be prevented.

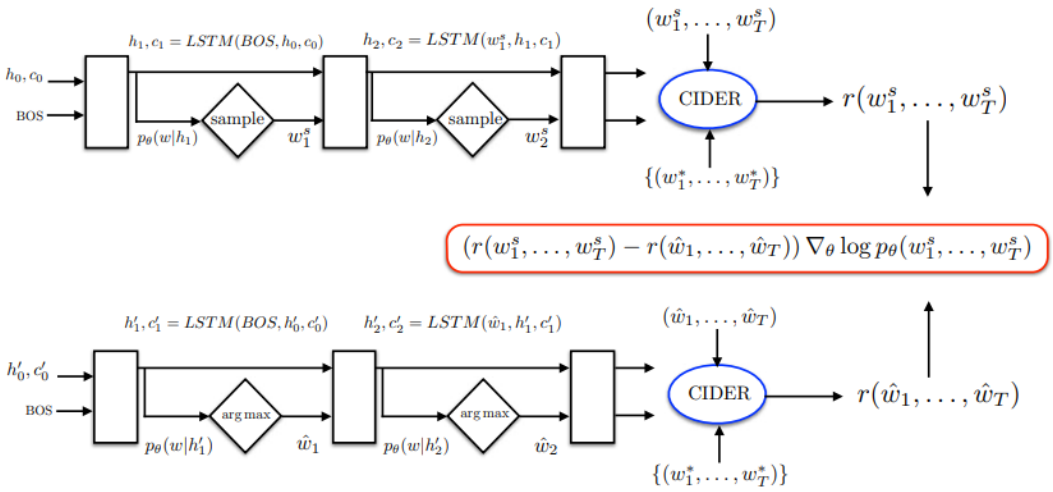


FIGURE 5.14: Self-critical sequence training (SCST). *Credit: (Rennie et al., 2017).*

Chapter 6

RL sentence compression

6.1 Introduction

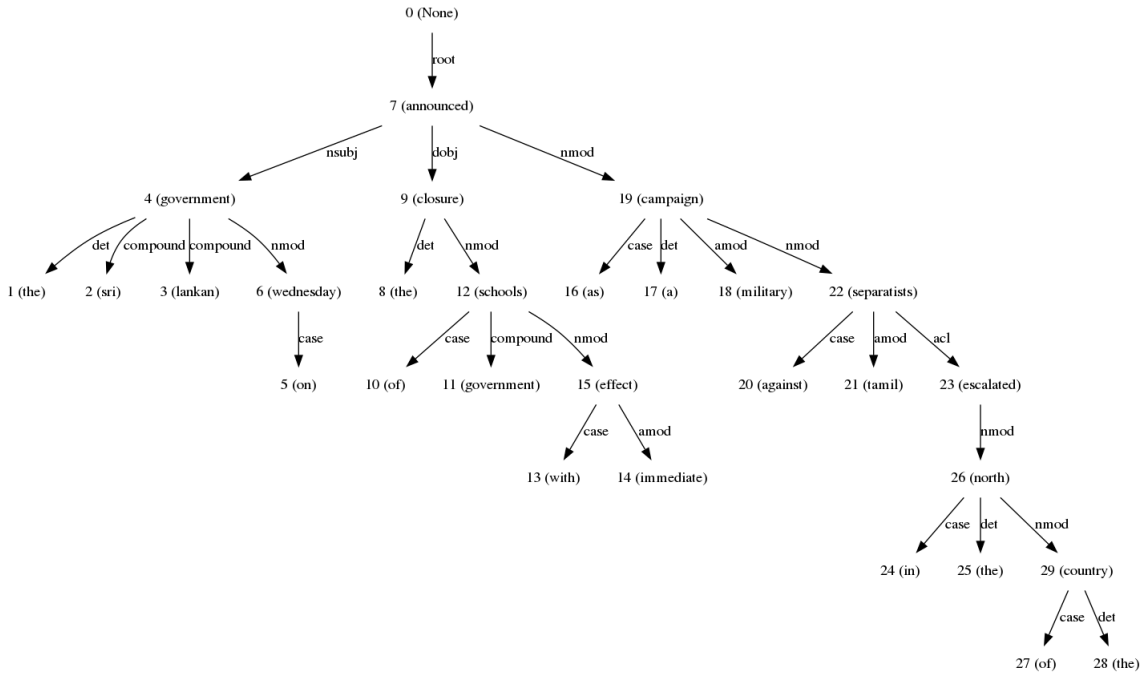
While previous work on sentence compression has often focused on extractive compression i.e., compressions where most of the words occurring in the short sentence version are also present in the corresponding input (Filippova and Altun, 2013; Filippova et al., 2015), the Gigaword corpus can be viewed as a corpus containing both extractive and abstractive compressions (or sentence summarization). Most of the previous studies on this dataset have only used dependency parsing as additional structural bias to the models (Cao et al., 2017; Amplayo, Lim, and Hwang, 2018; Li et al., 2018a; Song, Zhao, and Liu, 2018; Mihalcea and Tarau, 2004). However, subtrees from dependency structure are still not properly explored to help narrowing down input sentence into a short, concise and less ambiguous piece of information.

In this chapter, we propose a model that exploits syntactic parsing to extract coherent segments from the source document, then selects the best of these segments with reinforcement learning, and finally regenerates the summary with a recent sequence-to-sequence model. This model can thus transparently handles both types of extractive and abstractive summarizations. Furthermore, our approach departs from the recent end-to-end deep learning architectures by deterministically linearizing the syntactic tree into overlapping text segments to select with reinforcement learning. Although this method prevents a joint global optimization of the models, it also gives interesting adaptability and explainability properties. Adaptability, because it is easy to replace the syntactic parsing module with another shallow component, such as chunking, when full dependency parses are not available or not reliable. Explainability, because the segments that lead to the best summaries are clearly identified, which is not always the case with other deep learning approaches, for instance based on attention.

6.2 Extraction of Dependency Subtrees

We investigate different strategies for the extraction of the subtrees from the dependency structure of the input sentences. In all cases, we start from "sentence subtrees" i.e., subtrees which root node has an "nsubj" or an "nsubjpass" child node. Given such sentence trees, we then extract the following subtrees:

- **S-Tree**: All sentence trees.
- **1L:1R**: all subtrees of the sentence tree which contain one left- and one right-child. E.g., given the example in Figure 6.1, *"government announced closure"* and *"government announced campaign"*



S-tree, Depth 1	government anounced closure campaign
S-tree, Depth 2	sri lankan government on wednesday anounced schools closure military campaign separatists
1L:1R, Depth 1	government anounced closure government anounced campaign
1L:AR, Auxl, Depth 1	sri lankan government on wednesday anounced closure campaign

FIGURE 6.1: Dependency Tree and example subtrees extracted from the sentence *"The Sri-Lankan government on Wednesday anounced the closure of government schools with immediate effect as a military campaign against Tamil separatists escalated in the North of the country"* .

- **1L:AR (AL:1R)**: all subtrees of the sentence tree which contain one left- (right-) child and all right- (left-) children. E.g., *"government announced closure campaign"*
- **Auxl**: All subtrees below a subtree at depth 1 that contains "nsubj" or "nsubjpass".

We ignore all children nodes whose parent dependency relation is "punct" or "det". For each subtree type, we recursively extract subtrees of depth 1, 2 and 3. Figure 6.1 shows some examples of linearized extracted subtrees.

6.3 Model

Following Chen and Bansal, (2018), we use a two-steps model that combines an extractor agent to select dependency subtrees and an abstractor network to rewrite the extracted subtrees. Both networks are connected using reinforcement learning to overcome the non-differentiable behavior of the extractor and optimized with respect to the ROUGE evaluation, a standard metric used for sentence compression.

6.3.1 Extractor Network

Every linearized subtree from Section 6.2 is scored by the extractor network; this set of scored subtrees will later on be explored by the Reinforcement Learning agent to select the best candidates for summarization. In details, every subtree is passed to a temporal convolutional network (Kim, 2014) followed by a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to produce a subtree embedding h_j. Assuming that h_j is selected, it is passed to an LSTM decoder for generation, which outputs z_t. A pointer network (Vinyals, Fortunato, and Jaitly, 2015) computes a first attention e_t for z_t over all inputs $(h_j)_j$ with:

$$a_j^t = v_g^\top \tanh(W_{g1}h_j + W_{g2}z_t) \tag{6.1}$$
$$\alpha^t = \text{softmax}(a^t) \tag{6.2}$$
$$e_t = \sum_j \alpha_j^t W_{g1}h_j \tag{6.3}$$

It then computes the extraction probability of h_j with another attention:

$$\begin{aligned} u_j^t &= v_p^\top \tanh(W_{p1}h_j + W_{p2}e_t) \\ P(j_t|j_1,\ldots,j_{t-1}) &= \text{softmax}(u^t) \end{aligned} \tag{6.4}$$

All the W's and v's are trainable parameters. Similarly to (Chen and Bansal, 2018), we pretrain this extractor via a 'proxy' target label: for every ground-truth summary sentence, we find the most similar subtree via ROUGE-L$_{recall}$ metric and minimize this classification cross-entropy loss.

6.3.2 Abstractor Network

To generate compression, we use state-of-the-art sequence-to-sequence model with copy mechanism (See, Liu, and Manning, 2017). We also pretrain the abstractor by taking pairs of summaries and their extracted subtree (in Section 6.3.1). The decoder network is trained as usual as a language model:

$$L(\theta_{abs}) = -\frac{1}{M}\sum_{m=1}^{M} \log P_{\theta_{abs}}(w_m|w_{1:m-1})$$

6.3.3 Reinforce Extraction

To make an extraction agent, we use the vanilla policy gradient algorithm REINFORCE (Williams, 1992). At each extraction step t, the agent observes the current state $c_t = (D, d_{j_{t-1}})$, where $d \in D$: set of document sentence input. It samples an action $j_t \sim \pi_{\theta_a,\omega}(c_t, j) = P(j)$ from Eqn. 6.4 to extract a subtree and receives a reward $r(t+1) = ROUGE - L_{F_1}(g(d_{j_t}), s)$, where g is the abstracting function and s is the summary. We denote trainable parameters of the extractor agent as $\theta = \{\theta_a, \omega\}$ (in Section 6.3.1). Then, because vanilla REINFORCE yields high variance, we maximize this following policy gradient objective for the extractor:

$$\nabla_{\theta_a,\omega} J(\theta_a, \omega) = \mathbb{E}[\nabla_{\theta_a,\omega} \log \pi_\theta(c, j) A^{\pi_\theta}(c, j)] \tag{6.5}$$

where $A^{\pi_\theta}(c, j)$ is the *advantage function*, by definition:

$$A^{\pi_\theta}(c, j) = Q^{\pi_{\theta_a,\omega}}(c, j) - V^{\pi_{\theta_a,\omega}}(c)$$

The total return R_t is used as an estimate of action-value function $Q(c_t, j_t)$ and a critic network with trainable parameters θ_c is used to predict the state-value function $V^{\pi_{\theta_a,\omega}}(c)$. A baseline $b_{\theta_c,\omega}(c)$ is needed to reduce the variance of the estimation and helps to train the critic by minimizing this square loss:

$$L_c(\theta_c, \omega) = (b_{\theta_c,\omega}(c_t) - R_t)^2$$

This scheme is also referred as Advantage Actor-Critic (A2C), readers are invited to follow (Chen and Bansal, 2018) for more details. In a nutshell, RL extractor agent will favor the action of choosing a good candidate which leads to high ROUGE reward after the rewriting step of the abstractor and disfavor the reversed action which leads to the selection of bad candidates.

6.4 Evaluation

Data We evaluate our approach on the Gigaword corpus (Rush, Chopra, and Weston, 2015), a corpus of 3.8M sentence-headline pairs and where the average input sentence length is 31.4 words (in the training corpus) and the average sentence compression length is 8.3 words. The test set consists of 1951 sentence/compression pairs. Like Rush, Chopra, and Weston, (2015), we use 2000 sample pairs (among 189K pairs) as development set. The statistics of the Gigaword corpus is presented in Table 6.1.

Dataset	Train	Dev.	Test
Count	3.8M	189k	1951
AvgSourceLen	31.4	31.7	29.7
AvgTargetLen	8.3	8.3	8.8

TABLE 6.1: Data statistics for the English Gigaword.

Extractive vs. Abstractive Compression To better assess the impact of our approach on abstractive vs extractive compression, we divide the data (training, dev and test) into two parts: a part (extractive) where 80% of the tokens present in the compressed sentence are present in the input and another part (abstractive) which contains all other instances. According to that criteria, out of the 1951 test instances, 207 are extractive and 1744 abstractive. We also report the ROUGE metrics over the whole corpus, to allow for comparison with related works.

Evaluation Metric We adopt ROUGE (Lin, 2004) for automatic evaluation. It measures the quality of summary by computing overlapping lexical units between the candidate summary and actual summaries. We report ROUGE-1 (unigram), ROUGE-2 (bi-gram) and ROUGE-L (LCS) F1 scores. ROUGE-1 and ROUGE-2 mainly represent informativeness while ROUGE-L rather captures readability.

Hyperparameter Details Following (Chen and Bansal, 2018), we use the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001 for cross-entropy training of the extractor and abstractor. We use a learning rate of 0.0001 for extractor RL training. The vocabulary size is 30k, the batch size 32 samples, we use gradient clipping of 2.0 and early-stopping. We use 256 hidden units for all LSTM-RNNs. We truncate the maximum length of input sentences to 100 tokens and target sentences

to 30 tokens. ROUGE-recall is used to create label data for extraction phase as we want extracted sentences to contain as much information in the source as possible for paraphrasing. However, ROUGE-F_1 is used as reward for reinforce agent as the generation should be as *concise* as the gold target sentence.

6.4.1 Full Select-and-Paraphrase Model

Table 6.2 shows the results comparing the baseline model (a seq2seq model trained on input sentence/compression pairs) and our full Select-and-Paraphrase Reinforcement Learning (RL) model where the extractor is trained on all sentence trees (S-trees). The RL model under-performs (26.26 point R-L) the baseline (29.33 point R-L) because the extractor does not manage to handle the large number of candidate sub-trees (up to several hundreds). We thus explore next, using an oracle RL selector, which of the selection methods proposed in Section 6.2 help to reduce the set of candidate sub-trees while still preserving the relevant information from the input document.

Model	**Extractive Data**			**Abstractive Data**			**Whole corpus**		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
Baseline	59.57	31.28	57.87	27.55	10.16	25.94	30.95	12.40	29.33
S&P Model	54.38	28.13	52.42	24.48	8.49	23.15	27.65	10.57	26.26
(*)							37.04	19.03	34.46

TABLE 6.2: Baseline (Seq2Seq trained on Sentence/Compression Pairs) vs. RL Select-and-Paraphrase Model (trained on S-Tree Data).

We also report in Table 6.2 the performances obtained with state-of-the-art summarization systems. These figures come from Cao et al., (2018) **(*)**, which appears to be the best system on the Gigaword corpus reported in `http://nlpprogress.com/english/summarization.html`, as of May 2019.

6.4.2 Oracle Setting

In Table 6.3, we compare our model with an "oracle reinforcement learning" component that always chooses the best candidate subtree to pass to the abstractor. This allows us to study the impact of each sub-tree selection processes described in Section 6.2 and identify the ones that preserve relevant information to summarize. We also apply our approach with another shallow syntactic process, by replacing the dependency parser by the CoreNLP OpenIE tool [1], which extracts a set of subject-predicate-object triples.

OpenIE triples vs. Dependency Subtrees. We can first observe that scores are lower when taking as input OpenIE triples rather than Dependency subtrees. The results show that our specific S-tree heuristic rule outperforms OpenIE triples by +9, +7, +9 rouge-1,-2,-L respectively for extractive data, and +3, +1 and +3 points for abstractive data. OpenIE triples were in fact used by Cao et al., (2017) on the same task and same dataset to improve faithfulness i.e., to favour output that is semantically correct with respect to the input sentence. Given that Cao et al., (2017) achieved good scores on the Gigaword data and that S-trees outperform OpenIE triples, this suggests that S-Tree subtrees are potentially good alternatives to OpenIE triples.

[1] `https://stanfordnlp.github.io/CoreNLP/openie.html`

Model	Extractive Data			Abstractive Data			Whole corpus		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Baseline	59.57	31.28	57.87	27.55	10.16	25.94	30.95	12.40	29.33
Oracle									
OpenIE	51.21	24.1	48.7	27.35	9.24	25.68	29.88	10.82	28.12
S-Tree	60.52	31.21	57.29	30.61	10.69	28.77	33.78	12.88	31.80
1L:1R	46.33	19.15	43.6	22.63	5.84	21.03	25.14	7.25	23.43
1L:1R, Auxl	64.04	28.32	60.55	33.23	10.01	30.3	36.50	11.95	33.51
1L:AR, AL:1R	43.99	20.4	42	21.16	5.48	19.71	23.58	7.06	22.07
1L:AR, AL:1R, Auxl	62.57	32.02	58.98	30.61	10.69	28.77	34.00	12.95	31.98
S-Tree, 1L:1R, Auxl	68.95	36.34	65.49	38.95	13.9	35.54	42.13	16.28	38.72
S-Tree+	70.38	38.79	66.4	40	14.75	36.34	43.22	17.30	39.53

TABLE 6.3: Baseline vs. Oracle Results. The last row S-tree+ includes Stree, 1L:1R, Auxl, 1L:AR, AL:1R, Auxl

Extractive vs. Abstractive. Unsurprisingly, the impact of the input dependency subtrees is much larger on extractive data. For extractive compressions, the scores increase by roughly a factor of two suggesting that the match between input dependency subtrees and summaries is much larger for extractive than abstractive data. This is in line with previous works (Filippova and Altun, 2013; Filippova et al., 2015), which show that extractive compression can be found by searching in the parse tree of the input sentence for a spanning tree linking the content words of the compression. It also indicates that further improvements on the Gigaword dataset will require a better modeling of the paraphrases and variations occurring in abstractive compressions.

Auxiliary subtrees. We observe a large, significant improvement from (1L:1R) to (1L:1R, Auxl) and similarly, between (1L:AR, AL:1R) and (1L:AR, AL:1R, Auxl). In fact, this increase shows up systematically in all our experiments. This shows the importance of subtrees below the subject level, and that dependents and modifiers of these nodes often contain key information that is preserved in the compressed sentence.

Syntax helps. The combination of subtrees shows substantial improvement. Among all possible setup, the (S-Tree, 1L:1R, Auxl, 1L:AR, AL:1R, Auxl) combination obtains the best performance. It is respectively +10, +7, +9 rouge-1,-2-L points higher than the first heuristic rule and the baseline seq2seq model on the extractive set. On the abstractive set, it is +13, +4, +11 rouge-1,-2,-L points higher respectively.

Qualitative analysis. Table 6.4 shows examples of multiple subtrees retrieved from the input document as well as the summaries generated from them. We can see that normal sequence-to-sequence with attention and copy mechanism struggles to identify important information and produces loops and repetitions in the end. On the other hand, thanks to the dependency structure, the summaries generated from subtrees contain short and coherent sentences.

Source
fred west told the **truth** – and should be believed – **when he exonerated his wife** in the **murders** of ## young women before killing himself , a jury heard wednesday .
Subtrees
...
12. fred west believed when exonerated wife heard wednesday
13. fred west believed when exonerated murders heard wednesday
14. fred west believed when exonerated killing heard wednesday
15. fred west believed he exonerated wife heard wednesday
16. fred west believed he exonerated murders heard wednesday
17. fred west believed he exonerated killing heard wednesday
18. fred west believed a jury heard wednesday
...
Abstract
fred west told truth when he exonerated his wife of murder defense by unk unk
Full source generation
jury hears west tells truth to be believed to be believed
Subtrees generation
...
12. fred west says he 's exonerated
13. fred west says west nile murders
14. fred west says it was exonerated in killing of ##
15. fred west says he exonerated wife
16. fred west says he exonerated murders
17. fred west says he exonerated killing of killing
18. fred west s west virginia jury hears
...
Oracle
fred west says he exonerated wife (from subtree 15)

TABLE 6.4: Example of oracle and full source generation.

6.5 Conclusion

We have proposed a flexible select-and-paraphrase summarization model that decouples the syntactic analysis process from the generation component, hence enabling plugging-in and out various syntactic parsing modules. We have demonstrated this flexibility by seamlessly exploiting both a full-blown dependency parser and the shallow OpenIE triples extractor. The dependency parser giving better results, we have further proposed multiple heuristics to extract from the syntactic tree the most informative subtrees for the task of summarization and analyzed experimentally their potential. Compared to the state-of-the-art end-to-end deep learning systems, the proposed approach has another advantage, as it may more easily explain its generated summaries by presenting to the user the actual subtrees that have lead to the output sentence. Although this approach can theoretically handle both extractive and abstractive summarization, we show that it is particularly effective on extractive types of summaries, and that more work is still required to improve the generator component of this architecture.

Chapter 7

Enriching summarization with syntactic knowledge

7.1 Introduction

As mentioned in the related work, RL-based models have previously been used in text-to-text generation to address the exposure bias and loss mismatch issues encountered by models trained using cross-entropy (Paulus, Xiong, and Socher, 2018; Narayan, Cohen, and Lapata, 2018; Celikyilmaz et al., 2018; Pasunuru and Bansal, 2018). Similarly, while neural networks allow for features to be learned automatically, explicitly enriching the input with linguistic features was repeatedly shown to improve performance (Sennrich and Haddow, 2016; Li et al., 2017; Nallapati et al., 2016).

In this chapter, we explore the relative impact of these two approaches on sentence summarization. More precisely, we assess and compare the quality of the summaries generated by syntax-aware, RL-trained and combined models with regard to several qualitative aspects that strongly impact the perceived quality of the generated texts: number of repetitions, sentence length, distribution of part-of-speech tags, relevance and grammaticality. Specifically, using the standard Gigaword benchmark corpus, we compare and combine an RL self-critical sequence training (SCST) method with syntax-aware models that leverage part-of-speech (POS) tags and/or dependency information.

7.2 Models

We train and compare models that differ along two main dimensions: training (cross-entropy vs. RL) and syntax (with and without syntactic information).

7.2.1 Baseline

The baseline is a sequence-to-sequence model consisting of a bidirectional LSTM encoder and a decoder equipped with and an attention and a copy pointer-generator mechanism. We briefly describe the work of each component as follows:

Encoder. The source sentence is encoded using two recurrent neural networks (denoted *bi-RNN*) (Chung et al., 2014): one reads the whole sequence of words $x = (x_1, ..., x_m)$ from left to right and the other from right to left. This results in a forward and backward sequence of hidden states $(\overrightarrow{h_1}, ..., \overrightarrow{h_m})$ and $(\overleftarrow{h_1}, ..., \overleftarrow{h_m})$ respectively. The representation of each source word x_j is the concatenation of the hidden states $h_j = [\overrightarrow{h_j}, \overleftarrow{h_j}]$.

Decoder. A RNN is used to predict the target summary $y = (y_1, ..., y_n)$. At each decoder timestep, a multi-layer perceptron (MLP) takes as input the recurrent hidden state s_i, the previously predicted word y_{i-1} and a source-side context vector c_i to predict the target word y_i. c_i is the weighted sum of the source-side vectors $(h_1, ..., h_m)$. The weights in this sum are obtained with an attention model (Bahdanau, Cho, and Bengio, 2014), which computes the similarity between the target vector s_{i-1} and every source vector h_j.

Copy-Pointer. The attention encoder-decoder tends to ignore the presence of rare words in the source, which might be important especially for the summarization task. The Copy-Pointer (See, Liu, and Manning, 2017) enables to either copy source words via a pointer or generate words from a fixed vocabulary. A soft switch p_{gen} is learned to choose between *generating* a word from the vocabulary by sampling from the output distribution of the decoder, or *copying* a word from the input sequence by sampling from the attention distribution.

7.2.2 Integrating Syntax

Conventional neural summarization models only rely on the sequence of raw words and ignore syntax information. We include syntactic information in our summarization model using the hierarchical-RNN topology introduced by Li et al., (2017) and comparing three sources of information: POS tags (Postag), dependency relations (Deptag) and their combination (Pos+Deptag).

Figure 7.1 shows a graphical depiction of the Pos+tag model. In essence, each source of information (sequence of tokens, of POS tags, of dependency relations) is encoded using a bidirectional LSTM and each input token is then represented by the concatenation of the hidden-states produced by each information source considered. For instance, in the Postag model, the POS tag bi-LSTM takes as input a sequence of POS tags and outputs a sequence of hidden states $\left(hp_j = [\overleftarrow{hp_j}; \overrightarrow{hp_j}]\right)_{1 \leq j \leq m}$ similarly to the word bi-RNN. Each hp_j is then concatenated with the input word embeddings ew_j and passed on to the word bi-RNN. For the Deptag model, the input sequence to the Deptag bi-LSTM includes, for each input tokens, the name of the dependency relation that relates this token to its syntactic head (e.g., *nsubj* for the token "Mark" in the sentence shown at the top of Figure 7.1). The Deptag bi-LSTM then output a sequence of hidden states $\left(hd_j = [\overleftarrow{hd_j}; \overrightarrow{hd_j}]\right)_{1 \leq j \leq m}$ which are concatenated with the corresponding word embeddings ew_j and passed to the word bi-RNN. Finally, for the Pos+Deptag model, both the POS tag and the syntactic hidden states are concatenated with the words embeddings to give the final input vector $[\overleftarrow{ew_j}; \overrightarrow{ew_j}; \overleftarrow{hp_j}; \overrightarrow{hp_j}; \overleftarrow{hd_j}; \overrightarrow{hd_j}]$ which is passed on to the upper-level word bi-RNN.

7.2.3 Reinforcement Learning

Summarization as an RL problem Neural summarization models are traditionally trained using the cross entropy loss. Ranzato et al., (2016) propose to directly optimize Natural Language Processing (NLP) metrics by casting sequence generation as a Reinforcement Learning problem. As most NLP metrics (BLEU, ROUGE, METEOR,...) are non-differentiable, RL is appropriate to reach this objective. The parameters θ of the neural network define a *natural policy* p_θ, which can be used to predict the next word. At each iteration, the decoder RNN updates its internal state (hidden states, attention weights, copy-pointer weights...). After generating

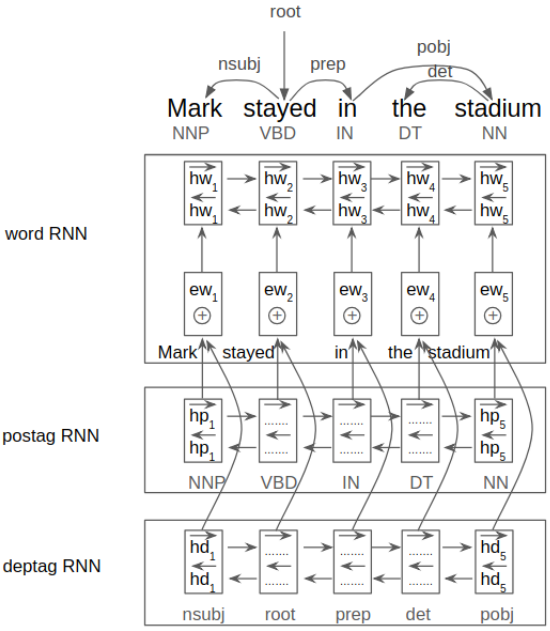


FIGURE 7.1: Pos+Deptag Model

the whole sequence, a reward $r(\cdot)$ is computed, for instance the ROUGE score. This reward is evaluated by comparing the generated sequence and the gold sequence. Similar to the previous Chapter 6, the RL training objective is to minimize the negative expected reward substracting the baseline to reduce the high variance of the estimation. The gradient of the learning objective is thus given as:

$$\nabla_\theta L^{RL}(\theta) = -\mathbb{E}_{w^s \sim p_\theta}\left[(r(w^s) - b)\nabla_\theta \log p_\theta(w^s)\right] \tag{7.1}$$

where $w^s = (w_1^s, \dots w_T^s)$ and w_t^s is the word sampled from the decoder at time step and the baseline can be an arbitrary function (function of θ or t), as long as it does not depend on w^s (Sutton and Barto, 1998).

Self-critical sequence training There are various ways to reduce RL variance and choose a proper baseline: for instance, using a second decoder (Ranzato et al., 2016) or building a *critic network* and optimizing with a *value network* instead of real reward (Bahdanau et al., 2016). In the following, we have chosen the self-critical sequence training (SCST) technique (Rennie et al., 2017), which has been shown to be very simple and effective. The main idea of SCST is to use, as baseline in the vanilla REINFORCE algorithm, the reward obtained with the inference algorithm used at test time. Equation 7.1 then becomes:

$$\nabla_\theta L^{RL}(\theta) = -\mathbb{E}_{w^s \sim p_\theta}\left[(r(w^s) - r(\hat{w}))\nabla_\theta \log p_\theta(w^s)\right] \tag{7.2}$$

where $r(\hat{w})$ is the reward obtained by the current model with the inference algorithm used at test time.

As demonstrated by Zaremba and Sutskever, (2015), we can rewrite this gradient formula as:

$$\frac{\partial L^{RL}(\theta)}{\partial s_t} = (r(w^s) - r(\hat{w}))(p_\theta(w_t|w_{t-1}^s, h_t) - 1_{w_t^s}) \tag{7.3}$$

where s_t is the input to the final softmax function in the decoder. The term on the right side resembles logistic regression, except that the ground truth w_t is replaced

by sampling w_t^s. In logistic regression, the gradient is the difference between the prediction and the actual 1-of-N representation of the target word:

$$\frac{\partial L^{XENT}(\theta)}{\partial s_t} = p_\theta(w_t|w_{t-1}, h_t) - 1_{w_t} \tag{7.4}$$

We see that samples that return a higher reward than $r(\hat{w})$ will be encouraged while samples that result in a lower reward will be discouraged. Therefore, SCST intuitively tackles well the exposure bias problem as it forces to improve the performance of the model with the inference algorithm used at test time. In order to speed up sequence evaluation at training time, we use greedy decoding with $\hat{w}_t = \arg\max_{w_t} p(w_t \mid h_t)$. [1]

Training objective and Reward The number of words in the vocabulary may be quite large in text generation, which leads to a large state space that may be challenging for reinforcement learning to explore. To reduce this effect, we follow Kryscinski et al., (2018) and adopt a final loss that is a linear combination of the cross-entropy loss and the policy learning loss:

$$L = (1 - \alpha)L^{XENT} + \alpha L^{RL} \tag{7.5}$$

α is a hyper-parameter that is tuned on the development set.

We use ROUGE-F_1 as the reward for the reinforce agent as the generation should be as concise as the gold target sentence.

7.3 Evaluation

We evaluated on the same Gigaword data and used the same automatic evaluation metric described in Section 6.4.

Implementation Our models implementations are based on the Fast-Abs-RL (Chen and Bansal, 2018) code [2]. Although this code is not optimized to give the best possible performances, it is flexible enough to allow for the integration of syntactic features. The hyperparameter α in Eq 7.5 needs careful tuning. Figure 7.2 illustrates a problematic case when α continuously increases until it reaches $\alpha = 1$ at iteration 10^5, where the RL models forget the previously learned patterns and degenerate. A good balance between exploration and supervised learning is thus necessary to prevent such catastrophic forgetting. We have found on the development set that the Reinforcement Learning weight α may increase linearly by (step/10^5) with the number of training iterations, until it reaches a maximum of 0.82 for the RL-s2s model and of 0.4 for the RL-s2s-syntax model. The main hyperparameters follow (Chen and Bansal, 2018) with number of part-of-speech tags 40, number of dependency tags 244 and arbitrarily 30 dimensions both for the part-of-speech and dependency embeddings. Our adapted code is published with an open-source licence at https://github.com/lethienhoa/Eval-RL.

[1] Two variants of this training method exist: TD-SCST and the "True" SCST, but both variants do not lead to significant additional gain on image captioning (Rennie et al., 2017). So we didn't explore these two variants for summarization as greedy decoding already obtains quite good result. We leave this for future work.

[2] https://github.com/ChenRocks/fast_abs_rl

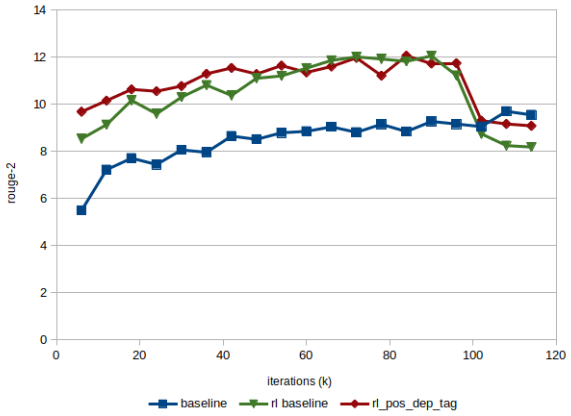


FIGURE 7.2: Catastrophic forgetting of the RL decoder on the Gigaword dev set

ROUGE Table 7.1 shows the performances measured in ROUGE score. State-of-the-art summarization system come from (Cao et al., 2018).

Models	#Params	Time for 1 epoch	R-1	R-2	R-L
Re3Sum (Cao et al., 2018)	-	-	37.04	19.03	34.46
Our Baseline s2s	6.617M	13h18m	27.57	10.29	26.02
Postag s2s	+312k	+54m	30.52	12.13	28.8
Deptag s2s	+318k	+1h13m	30.25	12.15	28.73
Pos+Deptag s2s	+630k	+1h34m	30.8	12.3	29.22
RL s2s	+0	-1h35m	29.94	11.64	28.54
RL postag s2s	-	-	30.82	12.19	29.12
RL deptag s2s	-	-	30.58	12.08	29.01
RL pos+deptag s2s	-	-	30.76	12.31	29.11

TABLE 7.1: Performance comparisons between models

Both Syntactic and RL models outperform the baseline. Syntax-aware models outperform the baseline by +1.84 (Postag s2s), +1.86 (Deptag s2s) and +2.01 (Dep+Postag s2s) ROUGE-2 points. While RL without syntax slightly under-performs syntactic models, it still achieves an improvement of +1.35 rouge-2 over the baseline. In other words, directly optimizing the ROUGE metric helps improve performance almost as much as integrating syntactic information. The combination of reinforcement learning with syntax information keeps increasing the score. However, the resulting improvement is smaller than when adding syntax without RL. We speculate that because the search space with syntax has a larger number of dimensions than without syntax, it may also be more difficult to explore with RL.

Parameters The baseline model has 6.617M parameters. This is increased by roughly 300K paramters for the Postag and the Deptag model and correspondingly by roughly 600K paramters for the Pos+Deptag model. In comparison, RL optimization does not involve any additional parameters. However, it requires two more decoder passes for the sampling and greedy predictions.

Speed Syntax-aware models are slightly longer to train than the baseline. Running on a single GPU GeForce GTX 1080, the baseline model requires 13h18m per

epoch with 114k updates while the training time of syntax-aware models increases by about 6% (Postag s2s). Also, it takes one week to get the pre-processing tag labels of these syntactic features for the whole 3.8M training samples of Gigaword corpus on 16 cores cpu machine Dell Precision Tower 7810. Surprisingly, adding the RL loss (which requires re-evaluating the ROUGE score for every generated text at every timestep) reduces training time by 12%. We speculate that the RL loss may act as a regularizer by smoothing the search space and help gradient descent to converge faster.

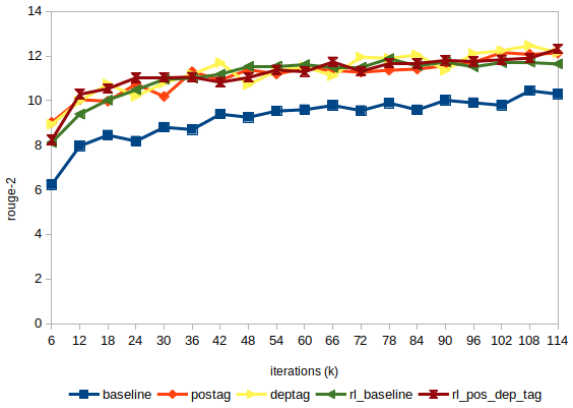


FIGURE 7.3: Evolution on test set during training

Learning Curve Figure 7.3 shows the evolution of Rouge-2 on the test set over 1 epoch. We can observe that syntactic models obtain a better performance than the baseline after the first 6k iterations. Sequence models with RL also quickly reach the same performance than syntactic models, even though the RL loss is only weakly taken into account at the start of training. As learning continues, the gap between the top models (with syntax and/or RL) and the baseline stays constant. The increased speed of training with RL, especially at the beginning of training, constitutes an important advantage in many experimental conditions.

7.4 Analysis

Repetitions Repetition is a common problem for sequence-to-sequence models (Tu et al., 2016; Sankaran et al., 2016). To assess the degree of repetitions in the generated summaries, we use Le et al., (2017)'s repetition rate metric which is defined as follows:

$$rep\_rat = \sum_{i=1}^{T(y)} \frac{1 + r(\widetilde{y}_i)}{1 + r(Y)} \tag{7.6}$$

where $\widetilde{y}_i$ and Y_i are the i^{th} generated sentence and i^{th} gold abstract target sentence respectively, and r is the number of repeated words: $r(X) = len(X) - len(set(X))$. $len(X)$ is the length of sentence X and $len(set(X))$ is the number of words that are not repeated in sentence X. Figure 7.4 compares the repetition rate of several models; the horizontal axis is the length of sentences, and the vertical axis is the repetition rate. The proposed RL-model combined with syntactic information performs the best on long sentences, with less repeated words than all other models. Indeed, short sentences are less likely to contain repetitions, but it is interesting to observe

that RL-training enriched with syntax improves the quality of long sentences on this aspect.

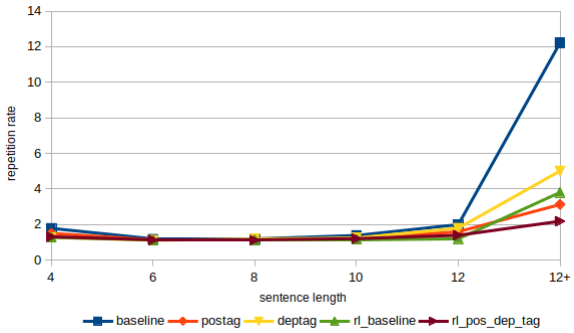


FIGURE 7.4: Repetition comparisons by length (lower is better)

Analysis by Postags To further investigate the linguistic structure of the generated output, we compute for each POS tag class T, the proportion of POS tags of type T relative to the number of generated words (on the test set). We group POS tags into 9 classes: cardinal numbers (CD), determiners (DT), nouns and proper nouns (NN), verbs (VV), adjectives (JJ), adverbs (RB), to (TO), prepositions and subordinating conjunctions (IN) and symbols (SYM). We evaluate whether the generated summary has a similar or different POS tags distribution than the ground truth by computing for each model the mean square error (MSE) between every generated and gold POS tag class. These errors are shown in Table 7.2. On average and for all POS tag classes, both syntax-aware and RL models are much closer (about 5 times) to the gold than the baseline. In a similar way as with repetitions, the best summarization model in terms of POS tag classes is the combined RL and syntax model.

Models	**Content words**				**Function words**					**MSE**
	NN	**VV**	**JJ**	**RB**	**CD**	**DT**	**TO**	**IN**	**SYM**	**to gold**
Gold target	49	12.5	12.9	1.6	1.3	1.5	2.5	10.6	4	0
Our baseline s2s	43.4	13.8	10.8	1.4	1.3	1.6	3.5	8.9	11.3	10.52
Postag s2s	50.4	14.5	12.1	1.3	1.3	1.1	3.9	8.1	2.1	2.07
Deptag s2s	49.8	14.5	12	1.7	1.3	1.1	3.7	8.9	1.9	1.59
Pos+Deptag s2s	51.4	14.7	12.6	1.6	1.1	1	3.7	7.9	1.8	2.72
RL s2s	50	14.1	11.9	1.3	1.6	1.2	4.1	9.1	1.6	1.71
RL pos+deptag s2s	49.9	14.3	12.4	1.3	1.5	1.2	3.6	9	2.2	1.28

TABLE 7.2: Proportion of generated postags

Effects on Long Sentences We group sentences of similar lengths together and compute the Rouge score. Figure 7.5 reports the Rouge-2 scores for various lengths of generated texts, with a 95% *t-distribution* confidence interval.

It shows that the RL and syntax models perform globally better than the baseline as sentences get longer. For long sentences (more than 10 words), this effect is more pronounced, the syntax(+RL) models outperform significantly the RL and baseline models.

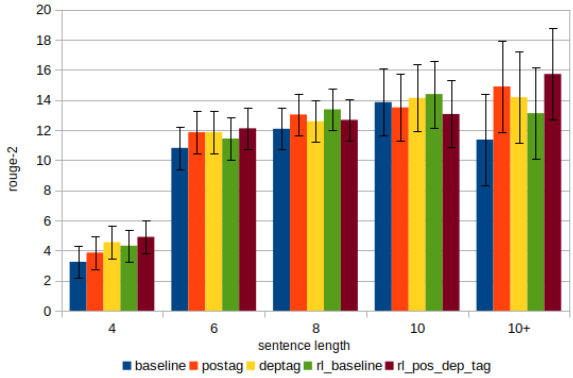


FIGURE 7.5: Performance comparisons by length

Human Evaluation In order to evaluate the quality of the summaries produced by the models, we asked 3 annotators to assess the relevance and grammaticality qualities of the summaries. Each criterion is rated with a score from 0 (bad) to 5 (excellent). Annotators are instructed to evaluate 50 samples randomly selected from the test set. The model information is anonymous to the annotators. The evaluation results with a 95% *t-distribution* confidence interval is shown in Table 7.3. We see that RL performs on par with postag, deptag on relevance and grammaticality criterions and they all outperfom baseline. This is consistent with the results on POS tag classes above which indicate that these models generate more content words and less function words than the baseline. Once again, RL with pos+deptag obtains the best result.

Models	**Rel.**	**Grammar.**
Baseline s2s	2.13 (+/-0.14)	2.47 (+/-0.18)
Postag s2s	3.26 (+/-0.18)	4.19 (+/-0.16)
Deptag s2s	3.17 (+/-0.19)	4.2 (+/-0.17)
RL s2s	3.23 (+/-0.19)	4.26 (+/-0.16)
RL pos+deptag	3.45 (+/-0.18)	4.49 (+/-0.13)

TABLE 7.3: Human Evaluations

Qualitative Analysis Table 7.4 shows some sample outputs of several models. Example 1 presents a typical repetition problem (the word *"atlantis"*) often found in the baseline. Both syntax and RL models manage to avoid repetitions. Example 2 shows that RL (without any syntactic information) can search and find surprisingly the same structure as the syntax-aware model. In the last example, the baseline fails as it stops accidentally after a modal verb while syntax and RL models can successfully generate well-formed sentences with subject-verb-object. However, semantically, RL and RL with pos+dep tag (like the baseline model) fail to capture the true meaning of the gold summary (*"transport plane"* instead of *"air force"* should be the real subject in this case). Deptag s2s seems the best in terms of summarizing syntactic and semantic content on these examples.

Source	the us space shuttle atlantis separated from the orbiting russian mir space station early saturday , after three days of test runs for life in a future space facility , nasa announced .
Abstract	atlantis mir part ways after three-day space collaboration by emmanuel unk
Baseline s2s	*atlantis atlantis atlantis separated from mir space station*
Postag s2s	us space shuttle atlantis separated from mir space station
Deptag s2s	atlantis separated from russian space station
Rl s2s	us shuttle atlantis separated from mir
Rl pos+deptag s2s	us shuttle atlantis separated from russian space station
Source	swedish telecommunications giant ericsson has reached a basic agreement to sell its relay production to japanese electronics company UNK corp , ericsson said thursday .
Abstract	ericsson sells relay production to unk 's unk corp
Baseline s2s	*ericsson to sell its its to sell its*
Postag s2s	ericsson to sell relay production to unk
Deptag s2s	ericsson reaches basic agreement to sell relay production
Rl s2s	ericsson reaches basic agreement to sell relay production
Rl pos+deptag s2s	ericsson sells relay production to unk corp
Source	the shooting down of the largest transport plane in the sri lankan air force has wrecked supply lines and slowed a major government offensive against the tamil rebel citadel of jaffna , analysts said .
Abstract	downing of plane slows sri lanka 's army onslaught on jaffna by amal jayasinghe
Baseline s2s	*sri lankan air force has*
Postag s2s	sri lankan air force plane shooting down
Deptag s2s	sri lanka 's largest transport plane shooting kills supply lines
Rl s2s	sri lankan air force wrecked supply lines
Rl pos+deptag s2s	sri lankan air force shooting down

TABLE 7.4: Generated examples of different models.

7.5 Conclusion

We have studied in details in this chapter the quality of the summaries that are generated by both syntactically-enriched summarization models and reinforcement-learning trained models, beyond the traditional ROUGE-based metric classically used to evaluate summarization tasks. We have thus focused on the quality of the generated summaries, in terms of the number of repeated words, which is a common issue with summarization systems, but also in terms of the distribution of various types of words (through their POS-tags) as compared to the gold. Because these aspects strongly depend on sentence length, we have also studied the impact of sentence length. Finally, we have manually evaluated the quality of the generated sentences in terms of relevance and grammaticality. Our results suggest that enriching summarization models with both syntactic information and RL training improves the quality of generation in all of these aspects. Furthermore, when computational complexity is a concern, we have shown that RL-only models may be a good choice because they provide nearly as good results as syntactic-aware models but with less parameters and faster convergence time.

Chapter 8

Conclusion

8.1 Summary and Conclusions

This thesis starts by presenting the latest advances in Neural Sentiment Analysis and Neural Text Summarization, since the rise of Deep Learning (Krizhevsky, Sutskever, and Hinton, 2012) and the first adoptions of neural methods for these two tasks in Socher et al., (2013) and Rush, Chopra, and Weston, (2015). Thus, Chapter 1 introduces the history and fundamentals of sentiment analysis and automatic summarization, as well as some of the limitations of the earlier methods, which have motivated the development of recent neural sentiment analysis and neural text summarization methods. In Chapter 2, we provide the necessary background to understand convolutional neural network architectures, which include window sliding and pooling operations, the core block for all current neural sentiment analysis methods. The focus is put on discussing the choice of the network architecture, the inclusion or not of external linguistic features, as well as the unit-level of representation of the inputs in Section 2.1. Next, in Section 2.2, we highlight the important relationship between sentiments and dialog acts, which has already been reported in the past but not carefully explored yet in the context of neural methods.

Chapter 3 begins our contribution. Since the pioneering work of Collobert and Weston, (2008), who have applied convolutional neural networks to multiple NLP tasks, the choice of the network architectures and the input-level for sentiment analysis varies much from explicit syntactic approach (Socher et al., 2013) on word-level input to learning-on-the-fly with convolutional nets on character input (Zhang, Zhao, and LeCun, 2015). Further, unlike in the computer vision domain, it is still unclear whether a similar benefit of increasing depths in neural networks for text classification may be expected, as both shallow-and-wide and deep convolutional models are evaluated on different tasks and datasets (Kim, 2014; Zhang, Zhao, and LeCun, 2015; Schwenk et al., 2017). This divergence in the community of research can lead to unnecessary duplicated efforts for practitionners to re-implement and fine-tune numerous convolutional models, and potentially obscure our understanding of the differences between respectively the image and text underlying mechanics. To avoid this problem, we carried a thorough analysis of different choices for the representation level of the inputs, between a shallow-and-wide network (Kim, 2014) and a very deep convolutional network, namely DenseNet (Huang et al., 2016), on five common text classification and sentiment analysis tasks. We chose to adapt the DenseNet and used it for the comparison and analysis as this model was shown to achieve state-of-the-art performances in image classification tasks. Our experiments revealed that a shallow-and-wide convolutional neural network at the word-level is still the most effective, and that deep models have not yet proven to be more powerful than shallow models for text classification tasks.

Chapter 4 summarizes our effort to fill the research gap in the correlation between sentiments and dialogs when modeling human interactions. Currently, the majority of works in neural sentiment classifiers still rely on raw text features, which are clearly not sufficient to modelize human opinions and behaviors. However, numerous studies (Pluwak, 2016; Clavel and Callejas, 2016; Boyer et al., 2011; Hasegawa et al., 2013) pointed out that close relations and strong respective influences exist between these two aspects. We have thus proposed to study a multi-task model for joint sentiment and dialog act recognition on social media. The results that we obtained are twofold: 1) transfer learning is quite efficient when the number of annotated labels for one task is smaller than for the other task. 2) though both tasks are not strongly correlated globally, there is enough mutual information to enable transfer learning. We have manually collected and annotated both dialogues and sentiments on the Mastodon social media corpus and distributed it freely to the community with an open-source license.

In addition to information about sentiments expressed in texts, another important information for users of NLP technologies is the semantic meaning contained in the documents itself. When only a few documents are concerned, then users can manually process them. But with the current speed at which data is accumulating, readers need technological assistance to handle this amount of information, such as provided by sentiment analysis and automatic summarization. To progress in this direction, the second part of the thesis is dedicated to automatic summarization, especially neural summarizers. Chapter 5 starts by discussing the drawbacks and limitations of sequence-to-sequence model, which have been naturally adapted from the domain of neural machine translation to the summarization task (Rush, Chopra, and Weston, 2015). These drawbacks include in particular incorrect, hallucinated and non-factual output. We walk the readers through various techniques to remedy this issue by integrating additional structural bias to make sure that the attention of the model spreads over key information in the source. Besides, we introduce in Section 5.1 another important line of research that aims at directly learning to distill out important sentences extracted from the long source document and making the neural sequence model more interpretable via reinforcement learning. Section 5.2 discusses some techniques used to enrich the neural summarizer with syntactic informations, and studies two problems faced by all generators, the exposure bias and loss mismatch, which can be solved by reinforcement learning based models.

Chapter 6 is our first contribution for sentence summarization. Although the models of Cao et al., (2017), Amplayo, Lim, and Hwang, (2018), Li et al., (2018a), Song, Zhao, and Liu, (2018), and Mihalcea and Tarau, (2004) are using dependency parses to fight hallucinations and non-factual output, they also make sequence-to-sequence systems' decisions harder to explain. The hybrid extraction-abstraction based model of Chen and Bansal, (2018) is better on this aspect but is only applied to summarization of long documents. While research on sentence-level summarization is very active but still needs explainability aspect, we propose a joint framework to progress in this direction by combining the best of both previous approaches. Our model works consecutively as: extracting coherent segments from the source document with the help of syntactic parsing, then selecting the best of these segments with reinforcement learning, and finally regenerating the summary with an advanced sequence-to-sequence model. The two main advantages of such a hybrid approach are that the model possesses better adaptability and explainability properties: 1) First, it is more flexible and integrates with various shallow syntactic parsing modules (chunking for ex.). 2) Second, the system is more interpretable as the segments that lead to the best summaries are clearly identified, while recent

sequence-to-sequence models only rely on attention mechanism for that.

In Chapter 7, we pursue our effort to qualify, from a syntactic point of view, the impact of reinforcement learning training for text generation. Reinforcement learning is becoming ubiquitous to enable training of summarization models with the ROUGE reward (Paulus, Xiong, and Socher, 2018; Narayan, Cohen, and Lapata, 2018) as well as with the SARI reward (Xu et al., 2016) for sentence simplification models (Zhang and Lapata, 2017). Besides, explicitly enriching the input with linguistic features is more costly with regard to preprocessing complexity and in terms of number of parameters, but this approach has also repeatedly been shown to improve the evaluation metric performance (Sennrich and Haddow, 2016; Li et al., 2017; Nallapati et al., 2016). Using the standard Gigaword sentence summarization task, we explore the quality of the summaries that are generated with both approaches. Specifically, we compare an RL self-critical sequence training (SCST) method with syntax-aware models that leverage POS tags and/or Dependency information. Through a range of qualitative criteria, such as: number of repetitions, sentence length, distribution of part-of-speech tags, relevance and grammaticality, we show that RL-only models already give nearly as good results as syntax-aware models with less parameters and faster training convergence. This is particularly helpful for cases when computational complexity is a concern.

8.2 Directions for Future Research

- *Impact of Neural Networks depth for sentiment analysis:* To verify the hypothesis that a deep convolutional neural network is necessary on textual data, we provide an adaptation of a very deep network, called DenseNet, for text classification and sentiment analysis tasks. However, particularly in the natural language processing domain, another class of successful deep networks, called the Transformer (Vaswani et al., 2017), is based on stacking multiple self-attention layers. It has been introduced after our study. This class of model, first proposed for neural machine translation, is later applied to numerous NLP tasks thanks to its superior performance. Letarte et al., (2018) propose an adaptation of this model and study the analysis of this model for sentiment analysis. They show that their Self-Attention Network (SANet), through modeling the interactions between all input word pairs, can provide more flexible and interpretable architecture than other deep learning methods for text classification. It would be interesting to extend the investigation of deep models in our thesis with this network. Further, while we have tried character and word as atom-level for the input feature representation, Sennrich, Haddow, and Birch, (2016) introduce a very effective approach based on byte pair encoding for word segmentation, which is capable of handling out-of-vocabulary words. They show that this representation can beat simple character bi-gram segmentation in their study. Thus, a combination of Transformer model with this feature input level is worth to explore next in our context for sentiment analysis.

- *Transfer learning between Sentiment Analysis and Dialog Act*: To better model sentiment analysis, we have proposed to train a multi-task hierarchical recurrent network on joint sentiment and dialog act recognition. Although these two tasks are not strongly correlated globally, we have nevertheless found a few specific patterns that exhibit a strong correlation to enable transfer learning. In computer vision, the intermediate hidden feature representations in deep

learning models have proven to be very beneficial when transfered to other tasks. For a long time, this good generalization phenomenon has not been demonstrated for textual data and is confirmed only partly again in our study. Recently, BERT (Devlin et al., 2019), a method to pretrain deep bidirectional representations from unlabeled text based on Transformers, ignites this hope for NLP. This method, which is based on language modeling as a source task, enables deep learning models to learn features with abundant language information and leads to numerous significant advances of transfer learning to many downstream tasks. In our context, it is worth to try this performant Transformer instead of our multi-task hierarchical recurrent network to jointly model sentiment and dialog act recognition. For modeling dialogue response generation, instead of predicting missing word, Chiang et al., (2019) already propose a Highway Recurrent Transformer to determine the most probable next sentence given the partial dialogue context. Following these studies, we could learn such a network as a pretraining task to finetune sentiment and dialog act recognition afterwards or directly predict the missing sentiment or missing dialog act. Through the success of BERT, we believe that more effort should be continued to put in these directions for transfer learning. Models should be studied to exploit free and already available labels from multiple different sources beside the sequence form and there's a need to continue to find such a good source of pretraining task beside language modeling.

- *RL extraction of syntax-based chunks for sentence compression*: The sequence-to-sequence summarization model is well known to generate hallucinations and unfaithful facts. In this thesis, we pursue the sentence summarization task by first exploiting segments from syntactic parsing and then performing selection of these segments by reinforcement learning before rewriting the summary with a sequence-to-sequence model. Though the coherence and the explainability properties of the framework are enhanced, we are aware of two major obstacles with this approach: 1) The number of segments from our defined rules results in a large selection space which makes reinforcement learning hard to deal with. 2) This current method is impossible to scale well with large document summarization. A recent study (Li, Thadani, and Stent, 2016) shows that segmenting EDUs (elementary discourse units) is good at conserving human-labeled summarization concepts. We speculate that these EDUs, which are not many in a sentence and are basically sub-clauses derived from sentence segmentation including dependencies such as clausal subjects and complements, can be a better alternative than our fine-grained analysis of syntactic segments for large document summarization under the reinforcement learning selection framework. Indeed, Cohan et al., (2018) already introduced a hierarchical encoder and an attentive discourse-aware decoder to summarize long documents. Xu et al., (2019) proposed a graph-based discourse-aware summarization model to avoid redundant or uninformative phrases in the generated summaries by popular methods based on sentence unit level. It is, therefore, worth to explore and compare the benefit of reinforcement learning in terms of concision, coherence and speed of learning with respect to these approaches.

Bibliography

Ain, Qurat Tul et al. (2017). "Sentiment Analysis Using Deep Learning Techniques: A Review". In:

Allahyari, Mehdi et al. (2017). "Text Summarization Techniques: A Brief Survey." In: *CoRR* abs/1707.02268. URL: http://dblp.uni-trier.de/db/journals/corr/corr1707.html#AllahyariPASTGK17.

Amplayo, Reinald Kim, Seonjae Lim, and Seung-won Hwang (2018). "Entity Commonsense Representation for Neural Abstractive Summarization". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 697–707. DOI: 10.18653/v1/N18-1064. URL: http://aclweb.org/anthology/N18-1064.

Austin, John Langshaw (1962). *How to do things with words*. William James Lectures. Oxford University Press. URL: http://scholar.google.de/scholar.bib?q=info:xI2JvixH8_QJ:scholar.google.com/&output=citation&hl=de&as_sdt=0,5&ct=citation&cd=1.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv e-prints* abs/1409.0473. URL: https://arxiv.org/abs/1409.0473.

Bahdanau, Dzmitry et al. (2016). "An Actor-Critic Algorithm for Sequence Prediction". In: *CoRR* abs/1607.07086. arXiv: 1607.07086. URL: http://arxiv.org/abs/1607.07086.

Banerjee, Satanjeev and Alon Lavie (2005). "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments". In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 65–72. URL: https://www.aclweb.org/anthology/W05-0909.

Banko, Michele et al. (2007). "Open Information Extraction from the Web". In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. IJCAI'07. Hyderabad, India: Morgan Kaufmann Publishers Inc., pp. 2670–2676. URL: http://dl.acm.org/citation.cfm?id=1625275.1625705.

Bengio, Samy et al. (2015). "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, pp. 1171–1179. URL: http://dl.acm.org/citation.cfm?id=2969239.2969370.

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent (2000). "A Neural Probabilistic Language Model." In: *NIPS*. Ed. by Todd K. Leen, Thomas G. Dietterich, and Volker Tresp. MIT Press, pp. 932–938. URL: http://dblp.uni-trier.de/db/conf/nips/nips2000.html#BengioDV00.

Bespalov, Dmitriy et al. (2011). "Sentiment classification based on supervised latent n-gram analysis." In: *CIKM*. Ed. by Craig Macdonald, Iadh Ounis, and Ian

Ruthven. ACM, pp. 375–382. ISBN: 978-1-4503-0717-8. URL: http://dblp.uni-trier.de/db/conf/cikm/cikm2011.html#BespalovBQS11.

Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) (2017). Vancouver, Canada: Association for Computational Linguistics. DOI: 10.18653/v1/S17-2. URL: https://www.aclweb.org/anthology/S17-2000.

Bing, Lidong et al. (2015). "Abstractive Multi-Document Summarization via Phrase Selection and Merging". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 1587–1597. DOI: 10.3115/v1/P15-1153. URL: https://www.aclweb.org/anthology/P15-1153.

Bollen, Johan, Huina Mao, and Xiaojun Zeng (2011). "Twitter mood predicts the stock market". In: *Journal of Computational Science* 2.1, pp. 1 –8. ISSN: 1877-7503. DOI: http://dx.doi.org/10.1016/j.jocs.2010.12.007. URL: http://www.sciencedirect.com/science/article/pii/S187775031100007X.

Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik (1992). "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: ACM, pp. 144–152. ISBN: 0-89791-497-X. DOI: 10.1145/130385.130401. URL: http://doi.acm.org/10.1145/130385.130401.

Boyer, Kristy et al. (2011). "An Affect-Enriched Dialogue Act Classification Model for Task-Oriented Dialogue". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 1190–1199. URL: https://www.aclweb.org/anthology/P11-1119.

Bridle, S. John (1990). "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition". In: 227–236.

Broder, Andrei Z. et al. (1997). "Syntactic Clustering of the Web". In: *Selected Papers from the Sixth International Conference on World Wide Web*. Santa Clara, California, USA: Elsevier Science Publishers Ltd., pp. 1157–1166. URL: http://dl.acm.org/citation.cfm?id=283554.283370.

Bunt, Harry et al. (2017). "Dialogue Act Annotation with the ISO 24617-2 Standard". In:

Cao, Ziqiang et al. (2017). "Faithful to the Original: Fact Aware Neural Abstractive Summarization". In: *CoRR* abs/1711.04434. arXiv: 1711.04434. URL: http://arxiv.org/abs/1711.04434.

Cao, Ziqiang et al. (2018). "Retrieve, rerank and rewrite: Soft template based neural summarization". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1, pp. 152–161. DOI: 10.18653/v1/p18-1015.

Celikyilmaz, Asli et al. (2018). "Deep Communicating Agents for Abstractive Summarization". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. New Orleans, Louisiana, pp. 1662–1675.

Cerisara, Christophe et al. (2018). "Multi-task dialog act and sentiment recognition on Mastodon". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 745–754. URL: https://www.aclweb.org/anthology/C18-1063.

Chen, Yen-Chun and Mohit Bansal (2018). "Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 675–686. DOI: 10.18653/v1/p18-1061. URL: http://aclweb.org/anthology/P18-1063.

Chiang, Ting-Rui et al. (2019). "Learning Multi-Level Information for Dialogue Response Selection by Highway Recurrent Transformer". In: *ArXiv* abs/1903.08953.

Chomsky, Noam (1957). *Syntactic Structures*. The Hague: Mouton and Co.

Chung, Junyoung et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". English (US). In: *NIPS 2014 Workshop on Deep Learning, December 2014*.

Clavel, Chloe and Zoraida Callejas (2016). "Sentiment Analysis: From Opinion Mining to Human-Agent Interaction". In: *IEEE Trans. Affect. Comput.* 7.1, pp. 74–93. ISSN: 1949-3045. DOI: 10.1109/TAFFC.2015.2444846. URL: http://dx.doi.org/10.1109/TAFFC.2015.2444846.

Cohan, Arman et al. (2018). "A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 615–621. DOI: 10.18653/v1/N18-2097. URL: https://www.aclweb.org/anthology/N18-2097.

Cohn, Trevor and Mirella Lapata (2008). "Sentence Compression Beyond Word Deletion". In: *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Manchester, UK: Coling 2008 Organizing Committee, pp. 137–144. URL: https://www.aclweb.org/anthology/C08-1018.

— (2009). "Sentence Compression As Tree Transduction". In: *J. Artif. Int. Res.* 34.1, pp. 637–674. ISSN: 1076-9757. URL: http://dl.acm.org/citation.cfm?id=1622716.1622733.

Collobert, Ronan and Jason Weston (2008). "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, pp. 160–167. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390177. URL: http://doi.acm.org/10.1145/1390156.1390177.

Collobert, Ronan et al. (2011). "Natural Language Processing (Almost) from Scratch". In: *J. Mach. Learn. Res.* 12, pp. 2493–2537. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.2078186.

Conneau, Alexis et al. (2016). "Very Deep Convolutional Networks for Natural Language Processing". In: *CoRR* abs/1606.01781. URL: http://arxiv.org/abs/1606.01781.

Core, Mark G. and James F. Allen (1997). "Coding Dialogs with the DAMSL Annotation Scheme". In: *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*. http://www.scientificcommons.org/43031741, pp. 28–35. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.7024.

Crook, Nigel, Ramón Granell, and Stephen G. Pulman (2009). "Unsupervised Classification of Dialogue Acts using a Dirichlet Process Mixture Model". In: *SIGDIAL Conference*.

Daumé Iii, Hal, John Langford, and Daniel Marcu (2009). "Search-based Structured Prediction". In: *Mach. Learn.* 75.3, pp. 297–325. ISSN: 0885-6125. DOI: 10.1007/s10994-009-5106-x. URL: http://dx.doi.org/10.1007/s10994-009-5106-x.

Dave, Kushal, Steve Lawrence, and David M. Pennock (2003). "Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews". In: *Proceedings of the 12th International Conference on World Wide Web*. WWW '03. Budapest, Hungary: ACM, pp. 519–528. ISBN: 1-58113-680-3. DOI: 10.1145/775152.775226. URL: http://doi.acm.org/10.1145/775152.775226.

Deriu, Jan et al. (2016). "SwissCheese at SemEval-2016 Task 4: Sentiment Classification Using an Ensemble of Convolutional Neural Networks with Distant Supervision". In: *SemEval@NAACL-HLT*.

Devlin, Jacob et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

Dhingra, Bhuwan et al. (2017). "Gated-Attention Readers for Text Comprehension." In: *ACL (1)*. Ed. by Regina Barzilay and Min-Yen Kan. Association for Computational Linguistics, pp. 1832–1846. ISBN: 978-1-945626-75-3. URL: http://dblp.uni-trier.de/db/conf/acl/acl2017-1.html#DhingraLYCS17.

Dodds, Peter Sheridan et al. (2011). "Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter". In: *CoRR* abs/1101.5120. URL: http://dblp.uni-trier.de/db/journals/corr/corr1101.html#abs-1101-5120.

Fan, Rong-En et al. (2008). "LIBLINEAR: A Library for Large Linear Classification". In: *J. Mach. Learn. Res.* 9, pp. 1871–1874. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1390681.1442794.

Filippova, Katja and Yasemin Altun (2013). "Overcoming the Lack of Parallel Data in Sentence Compression". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 1481–1491. URL: http://aclweb.org/anthology/D13-1155.

Filippova, Katja and Michael Strube (2008). "Sentence Fusion via Dependency Graph Compression". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '08. Honolulu, Hawaii: Association for Computational Linguistics, pp. 177–185. URL: http://dl.acm.org/citation.cfm?id=1613715.1613741.

Filippova, Katja et al. (2015). "Sentence Compression by Deletion with LSTMs". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 360–368. DOI: 10.18653/v1/D15-1042. URL: http://aclweb.org/anthology/D15-1042.

Forsythand, Eric N. and Craig H. Martell (2007). "Lexical and Discourse Analysis of Online Chat Dialog". In: *International Conference on Semantic Computing (ICSC 2007)*, pp. 19–26.

Gamon, Michael et al. (2005). "Pulse: Mining Customer Opinions from Free Text". In: *Proceedings of the 6th International Conference on Advances in Intelligent Data Analysis*. IDA'05. Madrid, Spain: Springer-Verlag, pp. 121–132. ISBN: 3-540-28795-7, 978-3-540-28795-7. DOI: 10.1007/11552253_12. URL: http://dx.doi.org/10.1007/11552253_12.

Gatt, Albert and Ehud Reiter (2009). "SimpleNLG: A Realisation Engine for Practical Applications". In: *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*. Athens, Greece: Association for Computational Linguistics, pp. 90–93. URL: https://www.aclweb.org/anthology/W09-0613.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: Proceedings of Machine Learning Research 9. Ed. by Yee Whye Teh and Mike Titterington, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

Go, Alec, Richa Bhayani, and Lei Huang (2009). "Twitter Sentiment Classification using Distant Supervision". In: *Processing*, pp. 1–6. URL: http://www.stanford.edu/~alecmgo/papers/TwitterDistantSupervision09.pdf.

Goldberg, Yoav and Graeme Hirst (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers. ISBN: 1627052984, 9781627052986.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2017). *Deep learning*. ISBN: 9780262035613 0262035618. URL: https://www.worldcat.org/title/deep-learning/oclc/985397543&referer=brief_results.

Hasegawa, Takayuki et al. (2013). "Predicting and Eliciting Addressee's Emotion in Online Dialogue". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 964–972. URL: https://www.aclweb.org/anthology/P13-1095.

He, Kaiming et al. (2016). "Deep Residual Learning for Image Recognition". In: pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Herzig, Jonathan et al. (2016). "Classifying Emotions in Customer Support Dialogues in Social Media". In: *SIGDIAL Conference*.

Hinton, Geoffrey et al. (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition". In: *Signal Processing Magazine*.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hu, Minqing and Bing Liu (2004). "Mining and Summarizing Customer Reviews". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '04. Seattle, WA, USA: ACM, pp. 168–177. ISBN: 1-58113-888-1. DOI: 10.1145/1014052.1014073. URL: http://doi.acm.org/10.1145/1014052.1014073.

Huang, Gao et al. (2016). "Densely Connected Convolutional Networks". In: cite arxiv:1608.06993Comment: CVPR 2017. URL: http://arxiv.org/abs/1608.06993.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: pp. 448–456. URL: http://jmlr.org/proceedings/papers/v37/ioffe15.pdf.

Jeong, Minwoo, Chin-Yew Lin, and Gary Geunbae Lee (2009). "Semi-supervised Speech Act Recognition in Emails and Forums". In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*. EMNLP '09. Singapore: Association for Computational Linguistics, pp. 1250–1259. ISBN: 978-1-932432-63-3. URL: http://dl.acm.org/citation.cfm?id=1699648.1699671.

Joachims, Thorsten (1998). "Text categorization with Support Vector Machines: Learning with many relevant features". In: *Machine Learning: ECML-98*. Ed. by Claire Nédellec and Céline Rouveirol. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 137–142. ISBN: 978-3-540-69781-7.

Joulin, Armand et al. (2016). "Bag of Tricks for Efficient Text Classification". In: cite arxiv:1607.01759. URL: http://arxiv.org/abs/1607.01759.

Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom (2014). "A Convolutional Neural Network for Modelling Sentences". In: *Proceedings of the 52nd Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Baltimore, Maryland: Association for Computational Linguistics, pp. 655–665. URL: http://www.aclweb.org/anthology/P14-1062.

Kaya, Mesut, Guven Fidan, and Ismail Hakki Toroslu (2013). "Transfer Learning Using Twitter Data for Improving Sentiment Classification of Turkish Political News." In: *ISCIS*. Ed. by Erol Gelenbe and Ricardo Lent. Vol. 264. Lecture Notes in Electrical Engineering. Springer, pp. 139–148. ISBN: 978-3-319-01603-0. URL: http://dblp.uni-trier.de/db/conf/iscis/iscis2013.html#KayaFT13.

Kim, Minkyoung and Harksoo Kim (2018). "Integrated neural network model for identifying speech acts, predicators, and sentiments of dialogue utterances". In: *Pattern Recognition Letters* 101, pp. 1–5.

Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: http://aclweb.org/anthology/D14-1181.

Kim, Yoon et al. (2016). "Character-aware Neural Language Models". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, pp. 2741–2749. URL: http://dl.acm.org/citation.cfm?id=3016100.3016285.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kiperwasser, Eliyahu and Miguel Ballesteros (2018). "Scheduled Multi-Task Learning: From Syntax to Translation". In: *Transactions of the Association for Computational Linguistics* 6, pp. 225–240. DOI: 10.1162/tacl_a_00017. URL: https://www.aclweb.org/anthology/Q18-1017.

Koehn, Philipp and Rebecca Knowles (2017). "Six Challenges for Neural Machine Translation". In: *Proceedings of the First Workshop on Neural Machine Translation*. Vancouver: Association for Computational Linguistics, pp. 28–39. DOI: 10.18653/v1/W17-3204. URL: https://www.aclweb.org/anthology/W17-3204.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

Kryscinski, Wojciech et al. (2018). "Improving Abstraction in Text Summarization". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 1808–1817. URL: https://aclanthology.info/papers/D18-1207/d18-1207.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 282–289. ISBN: 1-55860-778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655813.

Le, An Nguyen et al. (2017). "Improving Sequence to Sequence Neural Machine Translation by Utilizing Syntactic Dependency Information". In: *IJCNLP*.

Le, Hoa T., Christophe Cerisara, and Alexandre Denis (2018). "Do Convolutional Networks Need to Be Deep for Text Classification ?" In: *The Workshops of the The Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. Pp. 29–36. URL: https://aaai.org/ocs/index.php/WS/AAAIW18/paper/view/16578.

Le, Hoa T., Christophe Cerisara, and Claire Gardent (2019). "Quality of syntactic implication of RL-based sentence summarization". In: *arXiv e-prints*, arXiv:1912.05493, arXiv:1912.05493. arXiv: 1912.05493 [cs.CL].

Le, Hoa T., Christophe Cerisara, and Claire Gardent (2019). "RL Extraction of Syntax-Based Chunks for Sentence Compression". In: *Artificial Neural Networks and Machine Learning - ICANN 2019: Text and Time Series - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part IV*, pp. 337–347. DOI: 10.1007/978-3-030-30490-4_28. URL: https://doi.org/10.1007/978-3-030-30490-4_28.

LeCun, Y. et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. ISSN: 0018-9219. DOI: 10.1109/5.726791.

Letarte, Gaël et al. (2018). "Importance of Self-Attention for Sentiment Analysis". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, pp. 267–275. DOI: 10.18653/v1/W18-5429. URL: https://www.aclweb.org/anthology/W18-5429.

Li, Chenliang et al. (2018a). "Guiding Generation for Abstractive Text Summarization Based on Key Information Guide Network". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 55–60. DOI: 10.18653/v1/N18-2009. URL: http://aclweb.org/anthology/N18-2009.

Li, Haoran et al. (2018b). "Ensure the Correctness of the Summary: Incorporate Entailment Knowledge into Abstractive Sentence Summarization". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 1430–1441. URL: http://aclweb.org/anthology/C18-1121.

Li, Junhui et al. (2017). "Modeling Source Syntax for Neural Machine Translation". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 688–697. DOI: 10.18653/v1/P17-1064. URL: https://www.aclweb.org/anthology/P17-1064.

Li, Junyi Jessy, Kapil Thadani, and Amanda Stent (2016). "The Role of Discourse Units in Near-Extractive Summarization". In: *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Los Angeles: Association for Computational Linguistics, pp. 137–147. DOI: 10.18653/v1/W16-3617. URL: https://www.aclweb.org/anthology/W16-3617.

Lin, Chin-Yew (2004). "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Ed. by Stan Szpakowicz Marie-Francine Moens. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81. URL: http://www.aclweb.org/anthology/W04-1013.

Lin, Hui and Vincent Ng (2019). "Abstractive Summarization: A Survey of the State of the Art". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33, pp. 9815–9822. DOI: 10.1609/aaai.v33i01.33019815.

Liu, Bing (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers. ISBN: 1608458849, 9781608458844.

Luhn, H. P. (1958). "The Automatic Creation of Literature Abstracts". In: *IBM J. Res. Dev.* 2.2, pp. 159–165. ISSN: 0018-8646. DOI: 10.1147/rd.22.0159. URL: http://dx.doi.org/10.1147/rd.22.0159.

Maas, Andrew L. et al. (2011). "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT '11. Portland, Oregon: Association for Computational Linguistics, pp. 142–150. ISBN: 978-1-932432-87-9. URL: http://dl.acm.org/citation.cfm?id=2002472.2002491.

McCallum, Andrew and Kamal Nigam (1998). "A Comparison of Event Models for Naive Bayes Text Classification". In: *Learning for Text Categorization: Papers from the 1998 AAAI Workshop*, pp. 41–48. URL: http://www.kamalnigam.com/papers/multinomial-aaaiws98.pdf.

McDonald, Ryan et al. (2007). "Structured Models for Fine-to-Coarse Sentiment Analysis". In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, pp. 432–439. URL: https://www.aclweb.org/anthology/P07-1055.

Mihalcea, Rada and Paul Tarau (2004). "TextRank: Bringing Order into Text". In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, pp. 404–411. URL: https://www.aclweb.org/anthology/W04-3252.

Mikolov, Tomas et al. (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26*. Ed. by C.J.C. Burges et al., pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

Miyamoto, Yasumasa and Kyunghyun Cho (2016). "Gated Word-Character Recurrent Language Model". In: *EMNLP*. URL: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2016.html#MiyamotoC16.

Moreno, JMT (2014). *Automatic text summarization*. ISTE. Wiley. ISBN: 978-1-84821-668-6. URL: http://books.google.de/books?id=Q_2jBAAAQBAJ.

Nakov, Preslav et al. (2013). "SemEval-2013 Task 2: Sentiment Analysis in Twitter". In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA: Association for Computational Linguistics, pp. 312–320. URL: https://www.aclweb.org/anthology/S13-2052.

Nakov, Preslav et al. (2016a). "Developing a successful SemEval task in sentiment analysis of Twitter and other social media texts." In: vol. 50. 1, pp. 35–65. URL: http://dblp.uni-trier.de/db/journals/lre/lre50.html#NakovRKMKRSZ16.

Nakov, Preslav et al. (2016b). "Semeval- 2016 task 4: Sentiment analysis in twitter". In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pp. 1–18.

Nallapati, Ramesh, Feifei Zhai, and Bowen Zhou (2017). "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents". In: *Thirty-First AAAI Conference on Artificial Intelligence*.

Nallapati, Ramesh et al. (2016). "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, pp. 280–290. DOI: 10.18653/v1/K16-1028. URL: https://www.aclweb.org/anthology/K16-1028.

Narayan, Shashi, Shay B. Cohen, and Mirella Lapata (2018). "Ranking Sentences for Extractive Summarization with Reinforcement Learning". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans,

Louisiana: Association for Computational Linguistics, pp. 1747–1759. DOI: 10.18653/v1/N18-1158. URL: http://aclweb.org/anthology/N18-1158.

Nenkova, Ani and Kathleen R. McKeown (2011). "Automatic Summarization". In: *Foundations and Trends in Information Retrieval* 5.2-3, pp. 103–233. DOI: 10.1561/1500000015. URL: https://doi.org/10.1561/1500000015.

Nenkova, Ani and Rebecca Passonneau (2004). "Evaluating Content Selection in Summarization: The Pyramid Method". In: *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*. Boston, Massachusetts, USA: Association for Computational Linguistics, pp. 145–152. URL: https://www.aclweb.org/anthology/N04-1019.

Norouzi, Mohammad et al. (2016). "Reward Augmented Maximum Likelihood for Neural Structured Prediction". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., pp. 1723–1731. URL: http://papers.nips.cc/paper/6547-reward-augmented-maximum-likelihood-for-neural-structured-prediction.pdf.

Novielli, N. and C. Strapparava (2013). "The Role of Affect Analysis in Dialogue Act Identification". In: *IEEE Transactions on Affective Computing* 4.4, pp. 439–451. DOI: 10.1109/T-AFFC.2013.20.

Pasunuru, Ramakanth and Mohit Bansal (2018). "Multi-Reward Reinforced Summarization with Saliency and Entailment". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. New Orleans, Louisiana, pp. 646–653.

Paulus, Romain, Caiming Xiong, and Richard Socher (2018). "A deep reinforced model for abstractive summarization". In: *Proceedings of the 6th International Conference on Learning Representations*. Vancouver, BC, Canada.

Pluwak, Agnieszka (2016). "Towards Application of Speech Act Theory to Opinion Mining". In: pp. 33–44. DOI: 10.11649/cs.2016.004.

Prabha, M. I. and G. Umarani Srikanth (2019). "Survey of Sentiment Analysis Using Deep Learning Techniques". In: *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, pp. 1–9. DOI: 10.1109/ICIICT1.2019.8741438.

Qian, Qiao et al. (2015). "Learning Tag Embeddings and Tag-specific Composition Functions in Recursive Neural Network". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 1365–1374. DOI: 10.3115/v1/P15-1132. URL: https://www.aclweb.org/anthology/P15-1132.

Qureshi, Muhammad Atif, Colm O'Riordan, and Gabriella Pasi (2013). "Clustering with Error-Estimation for Monitoring Reputation of Companies on Twitter." In: *AIRS*. Ed. by Rafael E. Banchs et al. Vol. 8281. Lecture Notes in Computer Science. Springer, pp. 170–180. ISBN: 978-3-642-45067-9. URL: http://dblp.uni-trier.de/db/conf/airs/airs2013.html#QureshiOP13.

Radford, Alec, Rafal Jozefowicz, and Ilya Sutskever (2017). "Learning to Generate Reviews and Discovering Sentiment". In: cite arxiv:1704.01444. URL: http://arxiv.org/abs/1704.01444.

Radford, Alec et al. (2019). "Language Models are Unsupervised Multitask Learners". In:

Ranzato, Marc'Aurelio et al. (2016). "Sequence Level Training with Recurrent Neural Networks". In: *4th International Conference on Learning Representations, ICLR 2016,*

San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings. URL: http://arxiv.org/abs/1511.06732.

Ren, Yafeng et al. (2016). "Improving Twitter Sentiment Classification Using Topic-Enriched Multi-Prototype Word Embeddings". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Pp. 3038–3044. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11925.

Rennie, Steven J. et al. (2017). "Self-Critical Sequence Training for Image Captioning". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1195.

Ritter, Alan, Colin Cherry, and William B. Dolan (2010). "Unsupervised Modeling of Twitter Conversations". In: *HLT-NAACL*.

Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain". In: *Psychological Review*, pp. 65–386.

Rosenthal, Sara, Noura Farra, and Preslav Nakov (2017). "SemEval-2017 Task 4: Sentiment Analysis in Twitter". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 502–518. DOI: 10.18653/v1/S17-2088. URL: https://www.aclweb.org/anthology/S17-2088.

Rosenthal, Sara et al. (2014). "SemEval-2014 Task 9: Sentiment Analysis in Twitter". In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, pp. 73–80. DOI: 10.3115/v1/S14-2009. URL: https://www.aclweb.org/anthology/S14-2009.

Rush, Alexander M., Sumit Chopra, and Jason Weston (2015). "A Neural Attention Model for Abstractive Sentence Summarization". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 379–389. DOI: 10.18653/v1/D15-1044. URL: http://aclweb.org/anthology/D15-1044.

Sankaran, Baskaran et al. (2016). "Temporal Attention Model for Neural Machine Translation". In: *CoRR* abs/1608.02927.

Santos, Cícero Nogueira dos and Maira Gatti (2014). "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts." In: *COLING*. Ed. by Jan Hajic and Junichi Tsujii. ACL, pp. 69–78. ISBN: 978-1-941643-26-6. URL: http://dblp.uni-trier.de/db/conf/coling/coling2014.html#SantosG14.

Schank, R. and R. Abelson (1977). *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Hillsdale, NJ.: Lawrence Erlbaum Associates.

Schölkopf, Bernhard et al. (2001). "Estimating the Support of a High-Dimensional Distribution". In: *Neural Comput.* 13.7, pp. 1443–1471. ISSN: 0899-7667. DOI: 10.1162/089976601750264965. URL: https://doi.org/10.1162/089976601750264965.

Schwenk, Holger et al. (2017). "Very Deep Convolutional Networks for Text Classification." In: *EACL (1)*. Ed. by Mirella Lapata, Phil Blunsom, and Alexander Koller. Association for Computational Linguistics, pp. 1107–1116. ISBN: 978-1-945626-34-0. URL: http://dblp.uni-trier.de/db/conf/eacl/eacl2017-1.html#SchwenkBCL17.

See, Abigail, Peter J. Liu, and Christopher D. Manning (2017). "Get To The Point: Summarization with Pointer-Generator Networks". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics. DOI: 10.18653/v1/P17-1099. URL: https://www.aclweb.org/anthology/P17-1099.

Sennrich, Rico and Barry Haddow (2016). "Linguistic Input Features Improve Neural Machine Translation". In: *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, pp. 83–91. DOI: 10.18653/v1/W16-2209. URL: https://www.aclweb.org/anthology/W16-2209.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://www.aclweb.org/anthology/P16-1162.

Shawe-Taylor, John and Nello Cristianini (2004). *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press. ISBN: 0521813972.

Socher, Richard et al. (2011). "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions." In: *EMNLP*. ACL, pp. 151–161. ISBN: 978-1-937284-11-4. URL: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2011.html#SocherPHNM11.

Socher, Richard et al. (2012). "Semantic Compositionality through Recursive Matrix-Vector Spaces." In: *EMNLP-CoNLL*. Ed. by Jun'ichi Tsujii, James Henderson, and Marius Pasca. ACL, pp. 1201–1211. ISBN: 978-1-937284-43-5. URL: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2012.html#SocherHMN12.

Socher, Richard et al. (2013). "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Vol. 1631. Citeseer, p. 1642.

Song, Kaiqiang, Lin Zhao, and Fei Liu (2018). "Structure-Infused Copy Mechanisms for Abstractive Summarization". In: *CoRR* abs/1806.05658. arXiv: 1806.05658. URL: http://arxiv.org/abs/1806.05658.

Sordoni, Alessandro et al. (2015). "A Hierarchical Recurrent Encoder-Decoder For Generative Context-Aware Query Suggestion." In: *CoRR* abs/1507.02221. URL: http://dblp.uni-trier.de/db/journals/corr/corr1507.html#SordoniBVLSN15.

Stolcke, Andreas et al. (2000). "Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech". In: *Comput. Linguist.* 26.3, pp. 339–373. ISSN: 0891-2017. DOI: 10.1162/089120100561737. URL: https://doi.org/10.1162/089120100561737.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., pp. 3104–3112. URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Sutton, Richard S. and Andrew G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press. URL: http://www.cs.ualberta.ca/~sutton/book/the-book.html.

Tanaka, Hideki et al. (2009). "Syntax-Driven Sentence Revision for Broadcast News Summarization". In: *Proceedings of the 2009 Workshop on Language Generation and Summarisation (UCNLG+Sum 2009)*. Suntec, Singapore: Association for Computational Linguistics, pp. 39–47. URL: https://www.aclweb.org/anthology/W09-2808.

Tang, Duyu, Bing Qin, and Ting Liu (2015). "Deep Learning for Sentiment Analysis: Successful Approaches and Future Challenges". In: *Wiley Int. Rev. Data Min. and Knowl. Disc.* 5.6, pp. 292–303. ISSN: 1942-4787. DOI: 10.1002/widm.1171. URL: http://dx.doi.org/10.1002/widm.1171.

Tang, Duyu et al. (2014a). "Coooolll: A Deep Learning System for Twitter Sentiment Classification". In: *SemEval@COLING*.

Tang, Duyu et al. (2014b). "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification." In: *ACL (1)*. The Association for Computer Linguistics, pp. 1555–1565. ISBN: 978-1-937284-72-5. URL: http://dblp.uni-trier.de/db/conf/acl/acl2014-1.html#TangWYZLQ14.

Thelwall, Mike, Kevan Buckley, and Georgios Paltoglou (2012). "Sentiment strength detection for the social web". In: *Journal of the American Society for Information Science and Technology*.

Thelwall, Mike, Pardeep Sud, and Farida Vis (2012). "Commenting on YouTube videos: From guatemalan rock to El Big Bang." In: *JASIST* 63.3, pp. 616–629. URL: http://dblp.uni-trier.de/db/journals/jasis/jasis63.html#ThelwallSV12.

Tu, Zhaopeng et al. (2016). "Modeling Coverage for Neural Machine Translation". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 76–85. DOI: 10.18653/v1/P16-1008. URL: https://www.aclweb.org/anthology/P16-1008.

Turing, A. M. (1950). "Computing Machinery and Intelligence". English. In: *Mind*. New Series 59.236, pp. 433–460. ISSN: 00264423. URL: http://www.jstor.org/stable/2251299.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Venkatraman, Arun, Martial Hebert, and J. Andrew Bagnell (2015). "Improving Multi-step Prediction of Learned Time Series Models". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, pp. 3024–3030. ISBN: 0-262-51129-0. URL: http://dl.acm.org/citation.cfm?id=2888116.2888137.

Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly (2015). "Pointer Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, pp. 2692–2700. URL: http://dl.acm.org/citation.cfm?id=2969442.2969540.

Vinyals, Oriol et al. (2015). "Show and Tell: A Neural Image Caption Generator". In:

Vo, Duy-Tin and Yue Zhang (2015). "Target-Dependent Twitter Sentiment Classification with Rich Automatic Features." In: *IJCAI*. Ed. by Qiang Yang and Michael J. Wooldridge. AAAI Press, pp. 1347–1353. ISBN: 978-1-57735-738-4. URL: http://dblp.uni-trier.de/db/conf/ijcai/ijcai2015.html#VoZ15.

Vosoughi, Soroush and Deb Roy (2016). "Tweet Acts: A Speech Act Classifier for Twitter". In: *ICWSM*.

Wang, Xingyou, Weijie Jiang, and Zhiyong Luo (2016). "Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts." In: *COLING*. Ed. by Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad. ACL, pp. 2428–2437. ISBN: 978-4-87974-702-0. URL: http://dblp.uni-trier.de/db/conf/coling/coling2016.html#WangJL16.

Williams, Ronald J. (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning*, pp. 229–256.

Winograd, Terry (1971). "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language". In:

Woodsend, Kristian and Mirella Lapata (2011). "Learning to Simplify Sentences with Quasi-Synchronous Grammar and Integer Programming". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh,

Scotland, UK.: Association for Computational Linguistics, pp. 409–420. URL: https://www.aclweb.org/anthology/D11-1038.

Woszczyna, Monika and Alexander H. Waibel (1994). "Inferring linguistic structure in spoken language". In: *ICSLP*.

Wu, Yonghui et al. (2016). "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144*.

Xiao, Yijun and Kyunghyun Cho (2016). "Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers." In: *CoRR* abs/1602.00367. URL: http://dblp.uni-trier.de/db/journals/corr/corr1602.html#XiaoC16.

Xu, Jiacheng et al. (2019). "Discourse-Aware Neural Extractive Model for Text Summarization". In: *ArXiv*.

Xu, Wei et al. (2016). "Optimizing Statistical Machine Translation for Text Simplification". In: *Transactions of the Association for Computational Linguistics* 4, pp. 401–415.

Yang, Zhilin, Ruslan Salakhutdinov, and William W. Cohen (2016). "Multi-Task Cross-Lingual Sequence Tagging from Scratch." In: *CoRR* abs/1603.06270. URL: http://dblp.uni-trier.de/db/journals/corr/corr1603.html#YangSC16.

Yogatama, Dani et al. (2017). "Generative and Discriminative Text Classification with Recurrent Neural Networks." In: *CoRR* abs/1703.01898. URL: http://dblp.uni-trier.de/db/journals/corr/corr1703.html#YogatamaDLB17.

Yosinski, Jason et al. (2014). "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., pp. 3320–3328. URL: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf.

Zaremba, Wojciech and Ilya Sutskever (2015). "Reinforcement Learning Neural Turing Machines". In: *CoRR* abs/1505.00521.

Zarisheva, Elina and Tatjana Scheffler (2015). "Dialog Act Annotation for Twitter Conversations". In: *SIGDIAL Conference*.

Zhang, Lei Johnny, Shuai Wang, and Bing Liu (2018). "Deep Learning for Sentiment Analysis : A Survey". In: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 8.

Zhang, Xiang, Junbo Zhao, and Yann LeCun (2015). "Character-level Convolutional Networks for Text Classification". In: cite arxiv:1509.01626Comment: An early version of this work entitled "Text Understanding from Scratch" was posted in Feb 2015 as arXiv:1502.01710. The present paper has considerably more experimental results and a rewritten introduction, Advances in Neural Information Processing Systems 28 (NIPS 2015). URL: http://arxiv.org/abs/1509.01626.

Zhang, Xingxing and Mirella Lapata (2017). "Sentence Simplification with Deep Reinforcement Learning". In: *Proceedings of EMNLP*.

Zhang, Ye and Byron Wallace (2015). "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification". In: cite arxiv:1510.03820. URL: http://arxiv.org/abs/1510.03820.