UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

On Graph-Based Approaches for Protein Function Annotation and Knowledge Discovery

THÈSE

présentée et soutenue publiquement le 23 April, 2021

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Bishnu Sarker

Composition du jury

Président :	Anne Boyer : Professeure, Université de Lorraine, France.
Rapporteurs :	Christine Brun : DR CNRS, Marseille, France Mohamed Elati : Professeur, Université de Lille, France
Examinateurs :	Anne Boyer : Professeure, Université de Lorraine, France Albert Montresor : Professeur, University of Trento, Italy
Encadrants :	David W. Ritchie (till sept 2019) : DR Inria, Nancy, France Marie-Dominique Devignes (from sept 2019) : CRHC CNRS, Nancy, France Sabeur Aridhi : MC Université de Lorraine, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503
Institut National de la Recherche en Informatique et en Automatique- INRIA

Mis en page avec la classe thesul.

Remerciements

First and foremost, I would like to acknowledge the blessings of the Goddess Saraswati - the goddess of knowledge, music, art, speech, wisdom, and learning, the invisible divine energy that I believe to have omnipresence throughout my journey.

This journey would not have been possible without the generous funding from Inria Cordi-S doctoral grant. I would like to acknowledge the support from Inria Grand-Est. I also would like to thank DrEAM-LUE mobility grant for supporting my research visit to University of Montreal and Mila-Quebec Institute of Artificial Intelligence in Canada. I would like to express my gratitude to Capsid and Loria for hosting me in the lab, and to the University of Lorraine for enrolling me as a doctoral student.

I am extremely grateful to my wonderful supervisors, David W. Ritchie, Marie-Dominique Devignes, and Sabeur Aridhi for their continuous support, guidance, and caring. I can proudly say that I had the best advisors. Until his sudden departure from earth, Dave has been an amazing support for me. In the very first day, he picked me from the train station, drove me to the new home, handed over me the key, and payed for the rent until I received the salary. I never had to worry about anything other than research. Even regarding research, I had quality time with him. He was always there whenever I needed to discuss something. He was a wonderful human being with a great heart. May God rest his soul in peace.

After Dave's departure, Marie-Dominique took over the charge of supervising me for the rest of the period. She has been the biggest support during the period of pandemic. She has supported me like a mother during my the crisis moment. In every meeting she always asked me about my family, my wife, and me. She helped me to secure funding extension to support my research ideas during pandemic. This has been a great relief for me and helped me to focus on work.

Sabeur is a very kind person. I share a very friendly relationship with him. He always listened to my ideas attentively, and supported with his profound technical knowledge whenever possible. I would like to thank Sabeur for his support and kindness.

I am also grateful to Dr. Guy Wolf for accepting me as visiting researcher at University of Montreal. He was a wonderful researcher and very good human being.

Chapter 4 is a joint work with Navya Khare who has been an intern in the team working with me. I am very much grateful for her creative and spontaneous presence in the team, and making the project a success.

I would like to thank Malika Smail-Tabbone, Isaure Chauvot de Beauchene, Bernard Maigret for their pedagogic suggestions during tea time seminars and presentations.

I am lucky to have some wonderful colleagues like Kamrul Islam, Maria Elisa Ruiz Echartea, Athenais Vaginay for the good time we had spent together during this journey. My ideas and thinking has been sharpened by long discussion with Kamrul. I would like to convey my gratitude to Kamrul for his time and suggestions.

My beautiful wife Uma is a wonderful addition to my life. A pure soul to enlighten my heart. She has been integral part of this long journey. She left her dearest family, her job, and her comfortable life to accompany me in this journey in a new country. Her passionate and undivided love, encouragement, and sometimes scoldings kept me on the track. I am very much grateful to this wonderful lady, and I am happy that she is part of my life.

My life has been quite a journey. I was born in a lower income uneducated family in a poor village in Bangladesh. While managing food for a family was difficult, dreaming to educate a child supposed to be prohibited there. My father dared to have that dream, and successfully induced it into me. My Father used to be a carpenter. He barely passed the fifth grade. He lost his parents at a very early age, and raised by his elder brother. He had to leave school to join

his brother to work and earn livelihood from a very young age. But he dreamed to educate his children. I am whole heartily grateful to my father for igniting the passion into me to pursue higher education.

My father sacrificed his precious youth to ensure that I get the best education. He always pushed me by saying that "If needed, I will pay your fees by selling my blood, you don't need to worry about it. Just do your study". I firmly believe this has been the *Mantra* for me to coming this far. He became an immigrant worker in Singapore to support my education at the best college of the country, then to the best engineering university of the country despite the fact that he had to feed a family of seven persons. I, again, would like to express my heartiest gratitude to my father.

My mother is a symbol of ever struggling women. She has no formal education. Never been to school. She can not read and write. She fought a hard battle with poverty and ignorance. She used to wake up at 4.30 AM, prepares food for my father, then getting cattle and taking them to the field, collecting woods for cooking, taking care of kids, and then a tired body sleeps at midnight after feeding everyone at the dinner.Yet with uncertainty of what to eat next day. Me, my mother, and my father experienced the worst of the poverty. And, we share a wonderful journey to lift up a poor family to a middle income family. I could have get good job, and lived a good life right after my Masters. My mother could have lived a more relaxed life. She sacrificed her comfort for me to start this journey of Ph.D. My gratitude has no bounds for my mother.

I am also very much grateful to my in-laws (father-in-law, mother-in-law, and brother-in-law) for supporting me in this journey.

My siblings Liton, Juel, Shimul, Rakhi are the best possible siblings anyone could possibly have. I am very thankful to them for taking care of the family during my absence.

I would like to express my thanks to all of my friends and well-wishers that includes Cherry, Tulika, Rituraj, Ashmita, Zeba, Sunit, Gem, Robin, Subrata, Shubra who have been supporting from home and distance.

To Dr. David W. Ritchie whose kindness inspired me to become a good human being , To Uma, my lovely wife for her unconditional love, and sacrificial care,
To my parents, for bringing me on earth.

Sommaire

Table des figures

Première partie

Prologue

1

Introduction

1.1 Context

The recent advances in Artificial Intelligence(AI) have proved its tremendous potential to revolutionize discoveries in the field of biology and health. Along with the progress in Next Generation Sequencing (NGS) technologies, affordable genome sequencing has made it possible for AI technologies to find use cases in genomics and health. A plethora of sequences are already available in public databases. For example, UniProt KnowledgeBase [1] [The UniProt Consortium, 2015] the largest and most comprehensible public database for storing protein sequences contains more than 188 million sequences according to September 2020 release. This large volume of protein sequences opens up opportunities to perform analyses beneficial to answering long-hold questions in biology. On the other hand, it poses challenges due to the fact that this huge base of data is nearly impossible to annotate by manual effort. To put it in a specific context, we can continue with UniProtKB. UniProtKB is divided into two parts : 1) UniprotKB/SwissProt and 2) UniProtKB/TrEMBL.

In UniprotKB/SwissProt, the protein sequences are manually annotated or manually reviewed. This is a tremendous job for human annotators. It requires significant amount of time to read publications, to find the information regarding a particular protein, to identify functional properties and finally, to annotate it. The total process is costly as well. These are the primary reasons of UniProtKB/SwissProt having very slow growth over time. According to September 2020 release, there are roughly 560 thousands of protein sequences which are manually reviewed.

On the contrary, in UniProtKB/TrEMBL, the protein sequences do not have proper annotation or possibly they have annotation from automatic tools but, are not manually reviewed. When UniProtKB receives a new protein sequence, it puts it into UniProtKB/TrEMBL with minimum processing and the sequence is available online for further investigation. This is one of the reasons why UniProtKB/trEMBL has very sharp growth over the years. According to the release from September 2020, there are 188 million protein sequences in TrEMBL. However, without proper functional annotation, the use of the protein sequences is very limited.

To enrich and exploit this immensely valuable data, it is essential to annotate these sequences with functional properties such as Enzyme Commission (EC) numbers, Gene Ontology (GO) Annotation, for example. To reduce the gap between the annotated and unannotated protein sequences, it is essential to develop accurate automatic protein function annotation techniques.

1. https ://www.uniprot.org/

At present, two complementary systems are in action for automatic annotation of UniProtKB/TrEMBL sequences : 1) UniRule [Gattiker et al., 2003] is a rule-based system that uses manually engineered rules to assign appropriate annotation. Although rules in UniRule are generally very reliable, designing rules is a laborious and time consuming process and it works with low coverage. 2) SAAS (Statistical Automatic Annotation System) [Kretschmann et al., 2001] reduces the manual labour in UniRule system by automatically generating rules using the annotations of the SwissProt sequences and C4.5 decision tree algorithm [Quinlan, 1986].

More recently a number of tools have been introduced through CAFA [2] challenge. CAFA is a yearly challenge that seeks to find the annotation for a set of target protein sequences using automatic computation tools. The participants are free to use any resources that deemed useful to solve the problem. The recent report from CAFA consortium shows a heavy growth in the interest of using modern AI techniques specially deep learning techniques like convolutional neural network (CNN), recurrent neural network (RNN), long-short-term-memory(LSTM) etc. in solving protein function prediction problem. In most cases, the algorithm deals with sequences of proteins. Sequence is the primary protein data. However, with the help of well-known bioinformatics tools it is possible to identify information relating to motifs, pathways, interactions etc. While this adds an extra-step in the pipeline, it provides important information crucial to discovering protein-function association.

InterPro is a public database that accumulates domain, family and super-family information on proteins. It integrates data from different databases that use different aspects when computing the protein domains and motifs. InterPro domain architectures provide important insight regarding the functional characteristics of proteins. In this thesis we explore domain architecture of proteins by building domain similarity graph and performing neighborhood-based label propagation to infer annotation of un-annotated proteins. Learning representation has become an important research area in machine learning community. The main goal of the representation learning is to find meaningful low dimensional numerical vectors of real world entities. To take advantage of the recent advancement in representation learning in natural language processing, we applied word embedding techniques to embed InterPro domain in low dimensional vector space and using those embeddings, in later stage, we annotated proteins with EC number.

Along with InterPro signatures, UniProtKB lists many useful information relating to proteins and cross-references various data-sources. For example, for the proteins that are annotated manually, their UniProtKB/SwissProt entries contain information on pathway, post-translational-modificaton, taxon, GO annotation etc. Like-wise for un-reviewed proteins, there are still many related information regarding domain, taxon, genotypes-phenotypes etc. To make most of the available data, a natural way could be to build a relational network also known as Knowledge Graph (KG) that represents connections among biological entities along with the relation types. Knowledge graphs store edges as triples of the form (s,p,o) where s is head node or subject connected to the tail node or object o through the predicate or relation p. For example, (protein, has_domain, ipr) could be a possible edge in a hypothetical protein knowledge graph where a protein is connected to an *ipr* domain from the InterPro domain classification through a relation called *has_domain*. Knowledge graphs provide an effective way to explore knowledge and to infer new connections not readily seen in the graph. One of the challenges in knowledge discovery from knowledge graph is to compute the similarities among the entities. Knowledge graph embedding gives an effective way to compute similarities. The central idea of knowledge embedding is to embed or learn vector representation of entities and relations in a latent space. The similarity among the entities can then be computed as vector similarity. And, the task of protein

2. https ://www.biofunctionprediction.org/cafa/

function annotation can be reduced down to a task of link prediction in a knowledge graph. In this thesis, we propose a knowledge graph embedding model specifically designed to perform automatic protein function annotation from knowledge graph.

1.2 Contributions

— We present GrAPFI (Graph-based Automatic Protein Function Inference) for automatically annotating proteins with EC number functional descriptors from a protein-domain similarity graph. GrAPFI is a novel protein function annotation tool that performs automatic inference on a network of proteins that are related according to their domain composition. We validated the performance of GrAPFI using six reference proteomes in UniprotKB/SwissProt, namely Human, Mouse, Rat, Yeast, Arabidopsis thaliana and E.Coli. We also compared GrAPFI results with those of ECPred, DEEPre, and SVMProt as state-of-the-art EC prediction approaches using a benchmark dataset. We found that GrAPFI achieves better accuracy and comparable or better coverage with respect to these earlier approaches leveraging sequence homology.
The contribution is published as :

1. Sarker, B., Ritchie, D. W., Aridhi, S. (2018, December). Exploiting complex protein domain networks for protein function annotation. In International Conference on Complex Networks and their Applications (pp. 598-610). Cambridge, UK. Springer, Cham.
2. Sarker, B., Ritchie, D. W., Aridhi, S. (2020). GrAPFI : predicting enzymatic function of proteins from domain similarity graphs. BMC bioinformatics, 21, 1-15.

— We extend GrAPFI for automatic functional annotation of proteins using Gene Ontology (GO) terms. We include an efficient pruning and post-processing technique by integrating semantic similarity of GO terms computed from Gene Ontology. We observe from empirical results that the proposed hierarchical post-processing potentially improves the performance of GrAPFI and other GO annotation tools as well.
This contribution is published as :

1. Sarker, B., Khare, N., Devignes, M. D., Aridhi, S. (2020, May). Graph Based Automatic Protein Function Annotation Improved by Semantic Similarity. In International Work-Conference on Bioinformatics and Biomedical Engineering (pp. 261-272). Springer, Cham.
2. Sarker, B., Khare, N., Devignes, M. D., Aridhi, S. (2021, March). Improving Automatic GO Annotation With Semantic Similarity. BMC Bioinformatics (Under Review).

— We present an automatic EC annotation technique using neural network based word embedding exploiting domain and family information of proteins. In this experiment, we formulate the annotation task as a text classification task, and we use fastText [3], a library for learning word embeddings and text classification developed by Facebook's AI Research lab. We build a corpus of proteins using their domain architecture and learn fixed dimensional vector representation for proteins. We observe that the embeddings where we use domains acting as words perform much better than the embeddings with 3-mers (3 consecutive amino acid symbol) as words.
This contribution is published as :

3. https ://github.com/facebookresearch/fasttext

1. Sarker, B., Ritchie, D. W.,Aridhi, S. (2019, September). Functional Annotation of Proteins using Domain Embedding based Sequence Classification. In International Conference on Knowledge Discovery and Information Retrieval, Vienna, Austria.

— We introduce Prot-A-GAN : a generative knowledge graph embedding technique using GAN-like adversarial training for the purpose of protein function annotation. Following the terminologies of GAN : 1) we train discriminator with domain-adaptive negative sampling, and 2) we train generator as a random-walk over knowledge graph that identify path between protein and GO annotations. We formulate the problem of automatic protein annotation as a link prediction task where we want to infer that a protein is connected to an annotation depending on the type of relations they might hold. We observe that Prot-A-GAN performs with promising outcomes.

1.3 Thesis Synopsis

Chapter 2 presents background knowledge and state-of-the-art research in automatic protein function annotation as well as various machine learning tools applied to solve this problem. In chapters 3 and 4, we give a detailed description of GrAPFI for EC number and GO annotation respectively. Chapter 5 and 6 describe the application of representation learning for biological entities using sequence embedding and knowledge graph embedding respectively. The final chapter 7 draws the conclusion and perspectives of the work.

2

Background

This chapter will survey existing approaches for automatic function annotation of protein sequences. Because the two original characteristics of this thesis consist of using graph-based approaches and exploring domain composition of proteins, we will present in the first section the minimal necessary information for working with graphs, and in the second section the basic information about proteins, protein domains and the main vocabularies for functional annotation. The last section will discuss state-of-the-art researches in the field of automatic protein annotation, and knowledge graph embedding and generative adversarial networks.

2.1 Preliminaries About Graphs

2.1.1 Notations and Definitions

In this section, we first present some definitions and notations used in the paper.

Graph : A graph is a collection of objects denoted as $G = (V, E)$, where V is a set of vertices/nodes and $E \subseteq V \times V$ is a set of edges.

Weighted Graph : A weighted graph is a graph which is represented as a three tuple $G = (V, E, W)$ where :

— V is a set of nodes,
— $E \subseteq V \times V$ is a set of edges,
— W is a weight matrix where each cell W_{uv} represents a numerical weight of the edge $(u, v) \subseteq E$.

Labeled Graph : A labeled graph is a graph which is represented as a four tuple $G = (V, E, L, I)$ where :

— V is a set of nodes,
— $E \subseteq V \times V$ is a set of edges,
— L is a set of labels,
— $I : V \cup E \longrightarrow L$ is a labeling function.

Directed Graph : A Directed graph $G = (V, E)$ is a collection of objects where V is a set of vertices/nodes and $E \subseteq V \times V$ is a set of edges with ordered pair of vertices (u, v) such that $(u \longrightarrow v) \in E$.

Undirected Graph : An undirected graph is a collection of objects denoted as $G = (V, E)$, where V is a set of vertices/nodes and $E \subseteq V \times V$ is a set of edges with unordered vertices (u, v) such that if $(u \longrightarrow v) \in E$ exists then $(v \longrightarrow u) \in E$ must exist.

Neighbors : The neighbors of a node u are defined as $N(u) = \{v|(u, v) \in E\}$.

Degree : The degree of a node in a graph is the number of edges which touch it. The degree of a node u in a graph G is denoted $deg(u) = |N(u)|$.

Average Degree : The average degree of a graph $G = (V, E)$ is a measure of how many edges are in the set E compared to number of vertices in the set V. The average degree of a graph $G = (V, E)$ is defined by $Avgdeg = 2|E|/|V|$.

2.1.2 Homogeneous Graph

When all of the nodes $u \in V$ are of the same types, it is called homogeneous graph. For example, protein-protein interaction network is a homogeneous graph as all of the nodes in the graph are proteins and edges point to only the interaction partner without further mentioning the type of the interaction.

2.1.3 Heterogeneous Graph

A heterogeneous graph [Sun and Han, 2013] is a special kind of information network, which contains either multiple types of objects or multiple types of links. More formally, a heterogeneous graph, denoted as $G = (V, E)$, consists of an object set V and a link set E. To give an example, let us consider a network of proteins, their domain architecture, and functions. This network has different types of nodes : 1) proteins, 2) domain signatures, and 3) functions. The edges among the objects represent various types of associations depending which types of objects are linked together. Moreover, links can be of different types.

2.1.4 Knowledge Graph

A knowledge graph $G = \{\mathcal{V}, \mathcal{E}, \mathcal{R}, \mathcal{A}\}$ is a kind of directed heterogeneous graph where, $\mathcal{V}$ is a set of objects/entities of different types $\mathcal{A}$, $\mathcal{R}$ is the set of relation types that connect the objects in $\mathcal{V}$ and $\mathcal{E}$ is the set of edges represented as triple of the form (s, p, o), s : subject/head/source node, p : predicate/relation and o : object/tail/destination node. The entities of the knowledge graph are mapped to their corresponding node type with a mapping function $\phi : \mathcal{V} \longrightarrow \mathcal{A}$ and a link type mapping function $\psi : \mathcal{E} \longrightarrow \mathcal{R}$. Each entity $u \in \mathcal{V}$ belongs to an entity type $\phi(v) \in \mathcal{A}$, and each link $e \in \mathcal{E}$ belongs to a link type (relation) $\psi(e) \in \mathcal{R}$. In case of knowledge graphs, node and entity are used interchangeably to point the same thing. Node is more commonly used in homogeneous graphs whereas entity is commonly found in knowledge graph literature.

2.2 Preliminaries on Proteins, Domains and Functions

2.2.1 Proteins

Proteins are important macro-molecules. Proteins form the basis of life and play vital role in all living organism throughout the entire life-cycle. Proteins perform various functions in our body that needs to be understood to understand life, disease processes and guiding drug discovery efforts to combat the diseases. According to central dogma of molecular biology, the genetic information encoded inside DNA is transcripted into RNA and from RNA it is translated into proteins (Figure 2.1) that finally act inside the cell. Proteins are the end products in the decoding process of genetic information.

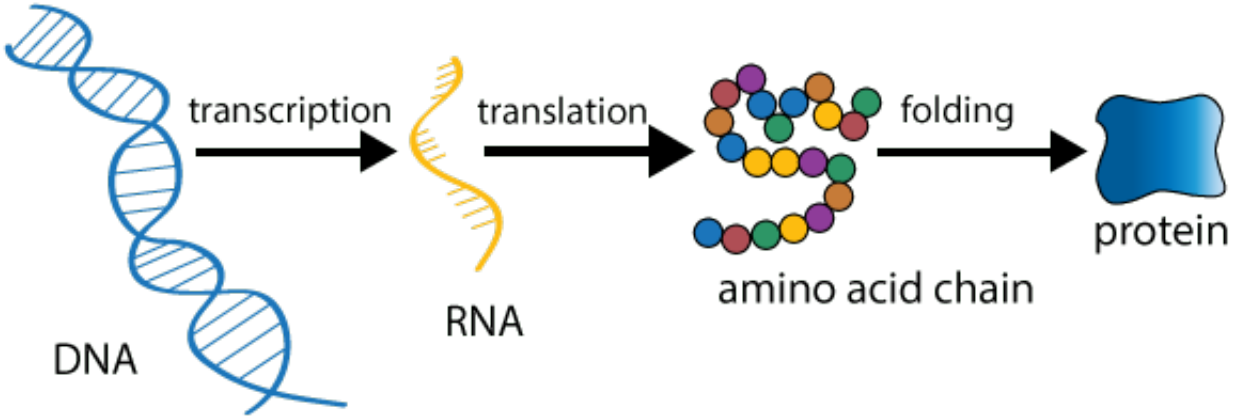


FIGURE 2.1 – Flow of genetic material from DNA to RNA to Proteins. Proteins are the end products in the decoding process of genetic information. Image source : https ://www.molecularecologist.com/2016/01/top-three-of-2015-melissa-debiasse/

2.2.2 Protein Sequence

Primarily proteins are found as sequences of amino acids, long chain of amino acids joined by peptide bonding. This is the primary structure for proteins. However, the 3D shape of the proteins is decided by their primary sequence structure. Proteins can be of various lengths. Computationally, proteins are long strings made from an alphabet of 20 letters symbolizing 20 amino acids. For example, following is a sequence of *Hemoglobin subunit gamma-2* protein. There are 147 letters in the sequence. Each letter in the sequence denotes one of 20 amino acids listed in Table 2.1.

MGHFTEEDKA TITSLWGKVN VEDAGGETLG RLLVVYPWTQ RFFDSFGNLS SASAIM-GNPK VKAHGKKVLT SLGDAIKHLD DLKGTFAQLS ELHCDKLHVD PENFKLLGNV LVTV-LAIHFG KEFTPEVQAS WQKMVTGVAS ALSSRYH

2.2.3 Protein Structure

Apart from the primary structure of a protein — its amino acid sequence - there are other three different forms that proteins can have. In other words, proteins can be found in four different forms ; 1) primary structure, linear chain of amino acids, 2) secondary structure, defines the distinctive three-dimensional structure from stable folding of patterns consists of alpha helics and beta sheets determined by the intramolecular bonding of the amino acid sequence, 3) tertiary structure, the ensemble of formations and folds in a single linear chain of amino acids, and 4) quaternary structure, the macromolecules with multiple polypeptide chains or subunits. The 3D structure of the protein *Hemoglobin subunit gamma-2* is shown in Figure 2.2.

2.2.4 UniProt Knowledgebase

The UniProt Knowledgebase (UniProtKB) [The UniProt Consortium, 2015] is currently the largest and most comprehensive resource for protein sequence and annotation data. It contains more than 188 million protein sequences according to September 2020 release. UniprotKB stores a number of secondary information derived from primary sequences extracted using bioinformatics software. UniProtKB is divided into two parts ; 1) UniprotKB/SwissProt, and 2) Uni-

Amino Acid	3-Letter Code	1-Letter Code	Chemical Formula
Alanine	ALA	A	$C_3H_7NO_2$
Arginine	ARG	R	$C_6H_{14}N_4O_2$
Asparagine	ASN	N	$C_4H_8N_2O_3$
Aspartic acid	ASP	D	$C_4H_7NO_4$
Cysteine	CYS	C	$C_3H_7NO_2S$
Glutamic acid	GLU	E	$C_5H_9NO_4$
Glutamine	GLN	Q	$C_5H_{10}N_2O_3$
Glycine	GLY	G	$C_2H_5NO_2$
Histidine	HIS	H	$C_6H_9N_3O_2$
Isoleucine	ILE	I	$C_6H_{13}NO_2$
Leucine	LEU	L	$C_6H_{13}NO_2$
Lysine	LYS	K	$C_6H_{14}N_2O_2$
Methionine	MET	M	$C_5H_{11}NO_2S$
Phenylalanine	PHE	F	$C_9H_{11}NO_2$
Proline	PRO	P	$C_5H_9NO_2$
Serine	SER	S	$C_3H_7NO_3$
Threonine	THR	T	$C_4H_9NO_3$
Tryptophan	TRP	W	$C_{11}H_{12}N_2O_2$
Tyrosine	TYR	Y	$C_9H_{11}NO_3$
Valine	VAL	V	$C_5H_{11}NO_2$

TABLE 2.1 – 20 Amino Acids along with their single letter and 3-letters codes.)

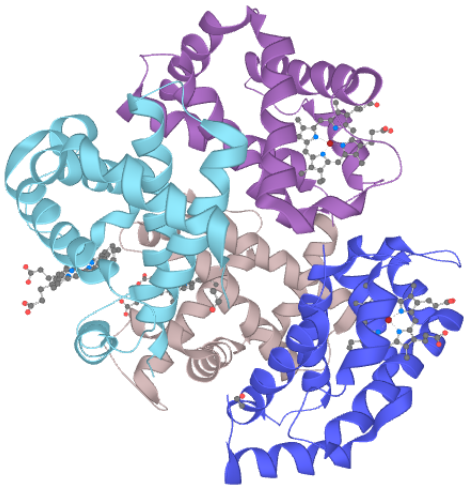


FIGURE 2.2 – 3D structure of *Hemoglobin subunit gamma-2* protein extracted from https ://www.uniprot.org/uniprot/P69892

ProtKB/TrEMBL.

In UniprotKB/SwissProt, the protein sequences are manually annotated or manually reviewed. This is a tremendous job for human annotator. It requires significant amount of time to read publications, to find the information regarding a particular protein, to identify functional properties, and to annotate it. The complete process is costly as well. Particularly, for UniprotKB/SwissProt data, proteins have functional annotations from Gene Ontology (GO) along with geneotype, phenotype, disease, and interactions data.

On the contrary, in the UniProtKB/TrEMBL, the protein sequences do not have proper annotation or possibly they have annotation from automatic tools and are not manually reviewed. When UniProtKB receives a new protein sequence, it put it into UniProtKB/TrEMBL with minimum processing and the sequence is available online for further investigation. This is one of the reasons that UniProtKB/trEMBL has very sharp growth over the years.

2.2.5 Proteins Family and Domains

Domains are the evolutionary information conserved in protein sequences. There are many computational tools that identify the domain information from protein sequence e.g. Pfam, gene3d, Prosite etc. A protein can have multiple domains and they are connected with protein by same relation type "protein :has_domain :ipr". Domains hold a hierarchical relationships among themselves. Therefore, domains themselves are sometimes connected using "is_a" relation to indicate the hierarchy.

2.2.6 InterPro Database

InterPro is a public database that accumulates domain, family and super-family information on proteins. InterPro provides functional analysis of proteins by classifying them into families and predicting domains and important sites [Blum et al., 2021]. It integrates data from different databases that use different aspects when computing the protein domains and motifs. InterPro integrates 13 protein signature databases into one central resource. The member databases are :

1. CATH-Gene3D [Sillitoe et al., 2019]
2. The Conserved Domains Database (CDD) [Lu et al., 2020]
3. HAMAP [Pedruzzi et al., 2015]
4. PANTHER [Mi et al., 2019]
5. Pfam [El-Gebali et al., 2019]
6. PIRSF [Nikolskaya et al., 2006]
7. PRINTS [Attwood et al., 2012]
8. PROSITE Patterns [Sigrist et al., 2012]
9. PROSITE Profiles [Sigrist et al., 2012]
10. SMART [Letunic and Bork, 2018]
11. the Structure–Function Linkage Database (SFLD) [Akiva et al., 2014]
12. SUPERFAMILY [Pandurangan et al., 2019] and
13. TIGRFAMs [Haft et al., 2012]

InterPro domain architectures provide important insight regarding the functional characteristics of proteins. Following are the important use cases of the InterPro database : (i) to identify

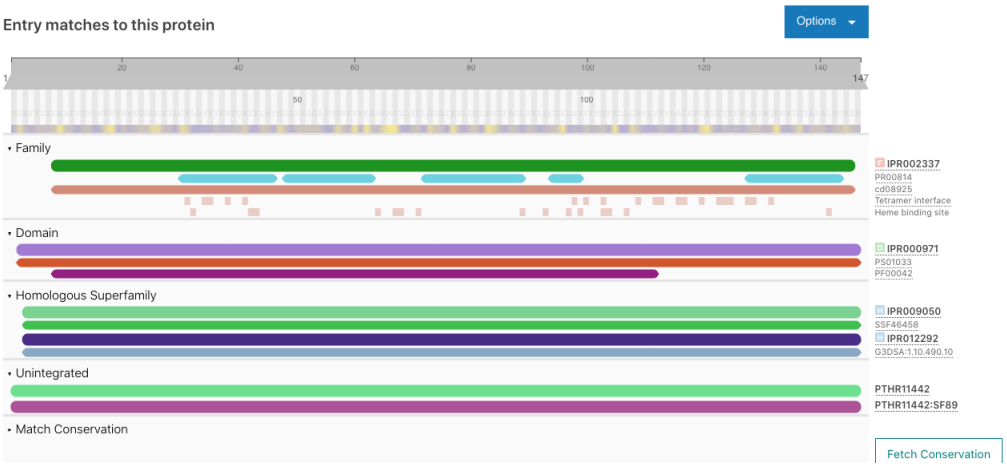


FIGURE 2.3 – Domain architecture of *Hemoglobin subunit gamma-2* generated from InterPro at https ://www.ebi.ac.uk/interpro/protein/UniProt/P69892/

what protein family a protein belongs to, (ii) what protein domains, sites or other features a protein contains, (iii) to annotate a genome with protein family information, and (iv) to annotate a genome with GO terms [Blum et al., 2021].

Figure 2.3 shows the domain architecture of *Hemoglobin subunit gamma-2* created using InterPro web resource at https ://www.ebi.ac.uk/interpro/protein/UniProt/P69892/. The figure portraits the alignment of various conserved sites in the protein sequence. The conserved sites are uniquely identified using InterPro signatures and provided with unique identifier.

2.2.7 Protein Functions

Proteins perform various functions. They act as enzyme, they participate in body's defense mechanism, they form structures, transport important chemicals etc. depicted in Figure 2.4.

— **Enzyme Commission Number** Enzymes are usually labelled following the Enzyme Commission (EC) system [Cornish-Bowden, 2014], the widely used numerical enzyme classification scheme. The EC System assigns each enzyme a four-digit number. This classification system has a hierarchical structure. The first level consists of the six main enzyme classes : (i) oxidoreductases, (ii) transferases, (iii) hydrolases, (iv) lyases, (v) isomerases and (vi) ligases, represented by the first digit. Each main class node further extends to several subclass nodes, specifying subclasses of the enzymes, represented by the second digit. Similarly, the third digit indicates the sub-subclass, and the fourth digit denotes the sub-sub-subclasses. Let us consider as an example a Type II restriction enzyme, which is annotated as EC 3.1.21.4. The first digit, 3, denotes that it is a hydrolase. The second digit, 1, indicates that it acts on ester bonds. The third digit, 21, shows that it is an endodeoxyribonuclease producing 5-phosphomonoesters. The last digit, 4, specifies that it is a Type II site-specific deoxyribonuclease.

— **Gene Ontology** Gene Ontology [Ashburner and et al., 2000] provides a hierarchical organization of controlled vocabulary reflecting the functional attributes of gene and gene products like proteins. GO annotation is assignment of GO terms to genes and proteins. For example,*Hemoglobin subunit gamma-2* is annotated with multiple GO terms indicating different functions such as Oxygen carrier activity (GO :0005344), Blood coagulation (0007596) etc. it is performing in our body shown in Figure 2.5.

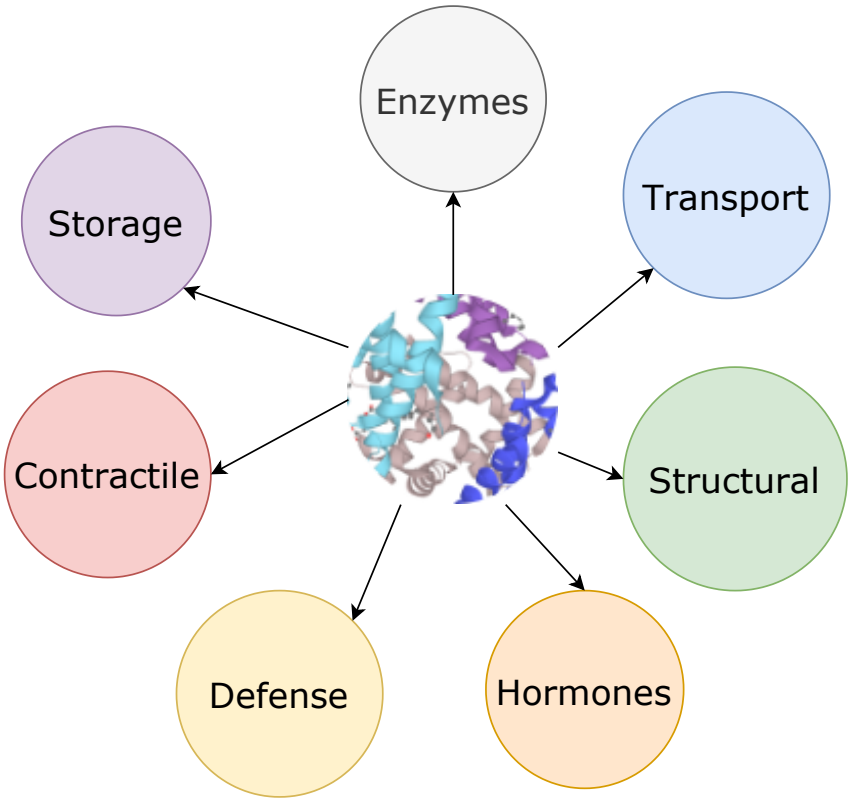


FIGURE 2.4 – Various functions of proteins.

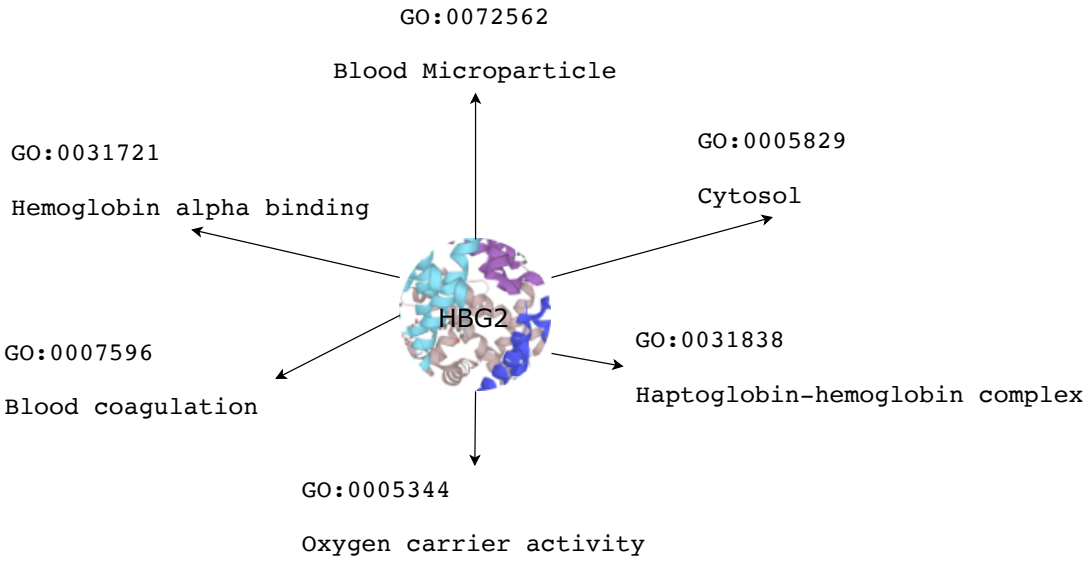


FIGURE 2.5 – Various functions of *Hemoglobin subunit gamma-2* proteins.

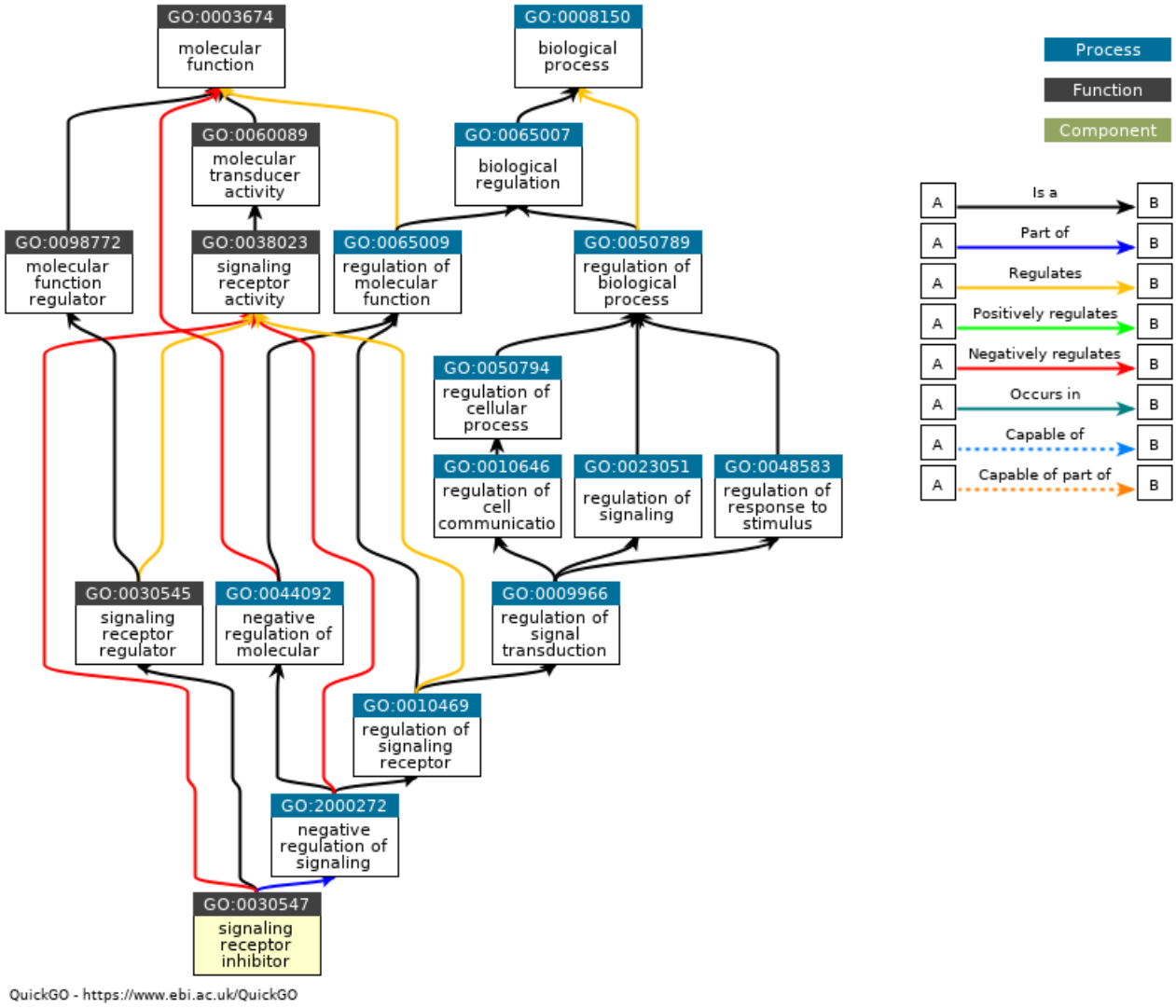


FIGURE 2.6 – The hierarchical organization of Gene Ontology (GO) terms extracted from https ://www.ebi.ac.uk/QuickGO/term/GO :0030547

In the gene ontology, GO terms are arranged hierarchically in three different directed acyclic graphs (DAG) namely, 1) Biological Process (BP), 2) Molecular Function (MF), and 3) Cellular Component (CC). Every node in a DAG represents a GO term and two connected GO terms are linked by different types of edges indicating different relationships. The most commonly used relationships are "is a", "part of", and "regulates". A small fraction of Gene Ontology is shown in the Figure 2.6.
Protein entries in UniProtKB are annotated using GO terms by using manual or computational approaches. GO annotations come with evidence codes that reflect the process involved in the annotation. There are 15 evidence codes that can be associated with a particular GO assignment. Following are the evidence codes used in UniProtKB GO annotations. A more detailed description can be found in http ://geneontology.org/docs/guide-go-evidence-codes/

1. Inferred from Experiment (EXP) is used to indicate that the functional association is

evident from experimental result.

2. Inferred from Biological Aspect of Ancestor (IBA) is used when function is derived from functional behavior of parent gene.
3. Inferred by Curator (IC) is used when annotation is inferred by curators from other GO annotations.
4. Inferred from Direct Assay (IDA) is used when function is derived from direct assay.
5. High-throughput Direct Assay (HDA)
6. Inferred from Electronic Annotation (IEA) is used when the annotation is performed through computation or automated transfer.
7. Inferred from Expression Pattern (IEP) Used when annotation is inferred from the timing or site of expression of a gene.
8. High-throughput Expression Pattern (HEP)
9. Inferred from Genomic Context (IGC) is used when the annotation is carried by available information about genomic context such as identity of neighboring genes, operon structure, and phylogenetic or other whole genome analysis.
10. Inferred from Genetic Interaction (IGI) is used to reflect the annotation evidence by "traditional" genetic interactions, such as suppressors and synthetic lethals, as well as other techniques, such as functional complementation, rescue experiments, or inferences about a gene drawn from the phenotype of a mutation in a different gene.
11. High-throughput Genetic Interaction (HGI)
12. Inferred from Mutant Phenotype (IMP) shows the variations or changes in a gene product, such as mutations or abnormal levels.
13. High-throughput Mutant Phenotype (HMP)
14. Inferred from Physical Interaction (IPI) is used to cover physical interactions between the gene product of interest and another molecule (ion, complex, etc.).
15. Inferred from Sequence Alignment (ISA) is used when the evidence is based on pairwise or multiple sequence alignment. That is the annotation is transferred based on the outcome of sequence alignment.
16. Inferred from Sequence Model (ISM) is used when the annotation is assigned using some kind of sequence modeling method (e.g. Hidden Markov Models).
17. Inferred from Sequence Orthology (ISO) is used when the assertion of orthology between the gene product and an experimentally characterized gene product in another organism is the main basis of the annotation.
18. Inferred from Sequence or Structural Similarity (ISS) is used to show that the annotation is performed based on sequence alignment, structure comparison, or evaluation of sequence features, such as composition.
19. Non-traceable Author Statement (NAS) is used when the annotation is found in publication and it can not be traced to original experiment.
20. Traceable Author Statement (TAS) is used when the annotation is found in publication and it can be traced to original experiment.

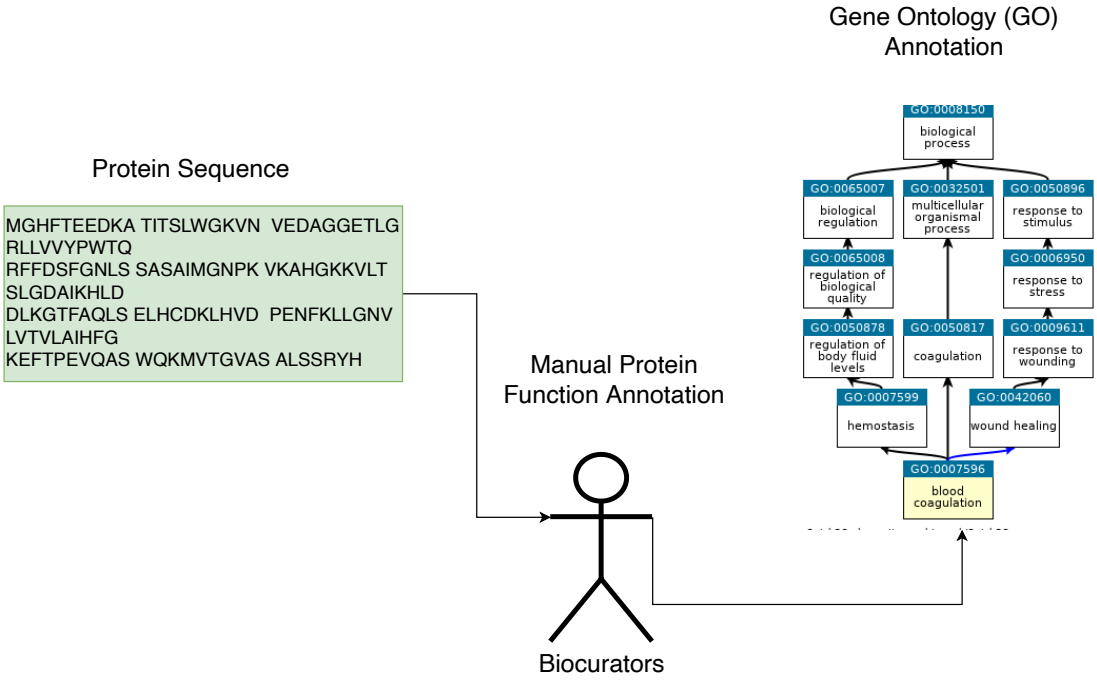


FIGURE 2.7 – Manual protein function annotation by biocurators.

2.2.8 Protein Function Annotation

Understanding the functional characteristics of proteins is vital because it provides understanding about life, disease progression, and drug discovery. The objective of protein function annotation is to assign appropriate functional characteristics to the proteins. Currently, both manual and computational pipelines are used to perform protein function annotation. As soon as the new protein sequence is selected, the annotation pipeline uses various bioinformatics tools, expert human knowledge, and scientific publications to understand the behavior of the protein, and perform the annotation with relevant characteristics using controlled vocabulary, for example, from EC system or Gene Ontology.

Manual Protein Function Annotation

Functional annotation of proteins by manual approach involves team expert biologists or biocurators selecting protein sequences based on special interest, applying bioinformatics tools to analyze them, reading scientific articles, retrieving relevant information, and finally, assigning annotation and performing quality control before storing into annotation database like UniProtKB/SwissProt.

In Figure 2.7, we see that biocurators take the protein sequence as input, and assign annotation using Gene Ontology, for example.

Manual annotation is a laborious job comprises six mandatory steps to be followed by the biocurators to ensure the consistency in the curation process as explained in UniProtKB [The UniProt Consortium, 2015] shown in Figure 2.8. A detailed description is provided by standard operating procedure published by UniProtKB consortium [4].

Following are the 6 steps of manual annotation of proteins :

4. https ://www.uniprot.org/docs/sop%5Fmanual%5Fcuration.pdf

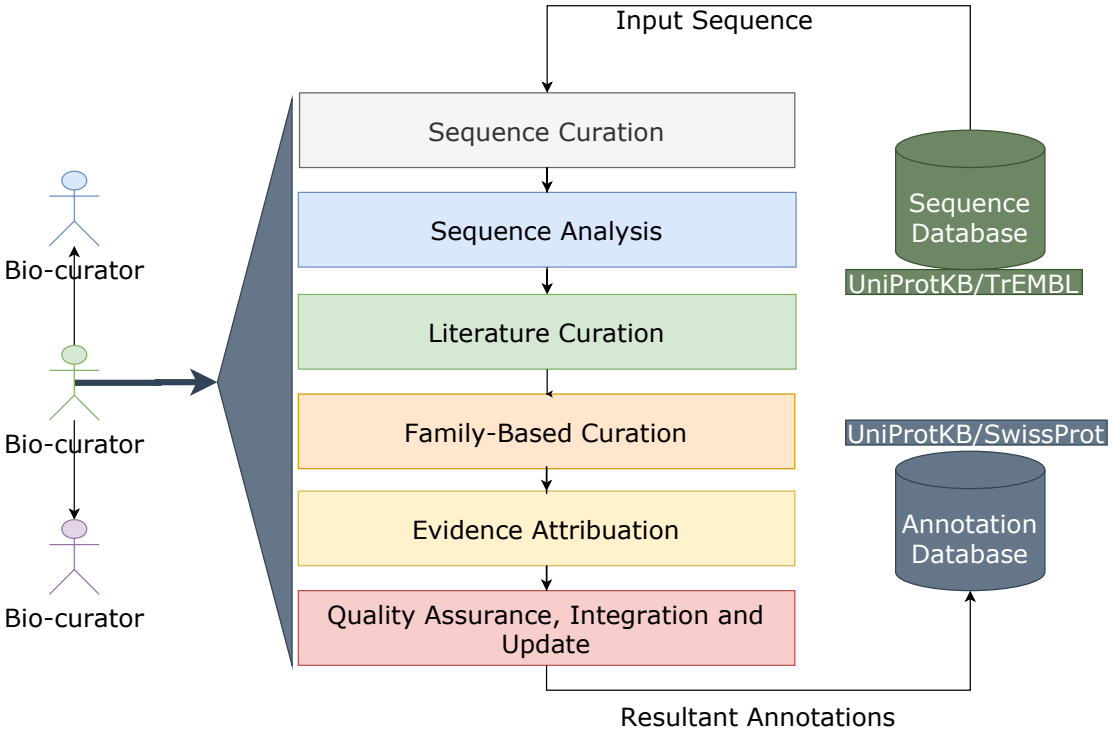


FIGURE 2.8 – Work-flow of manual protein function annotation

1. **Sequence curation**, upon selecting a protein sequence based curation priorities, biocurators run Blast searches [Altschul et al., 1997] against UniProtKB protein database to find the homologous sequences, and sequences from same gene. These sequences are analyzed together to reduce the redundancy in annotation process. The discrepancies among the sequences are recorded to further establish the consistency, accuracy, and quality of further curation steps of the annotation.
2. **Sequence analysis**, various bioinformatics tools are used in this step to identify post-translational modifications, sub-cellular location, transmembrane domains and protein topology, domain identification and protein family classification for the query protein. These results are manually reviewed before integration to the annotation pipeline.
3. **Literature curation**, an important step after sequence analysis is to find experimental evidences. Biocurators search literature databases like PubMed for relevant knowledge regarding the query protein. They read full papers, extract experimental findings and author statements, and update the current understanding of the query protein. This step is vital for finding GO annotation for the protein.
4. **Family-based curation**, the annotations gathered from the previous steps are propagated to the homologous sequences for attaining the consistancy in annotation of sequences from same family.
5. **Evidence attribution**, as it is evident from the previous steps, the annotations are coming from different sources and tools. Therefore, it is essential to record which source has been used for a particular annotation. In this step, sources of the annotations are indicated using some kind of evidence codes.
6. **Quality assurance and integration**, the final step is to ensure the quality, and to

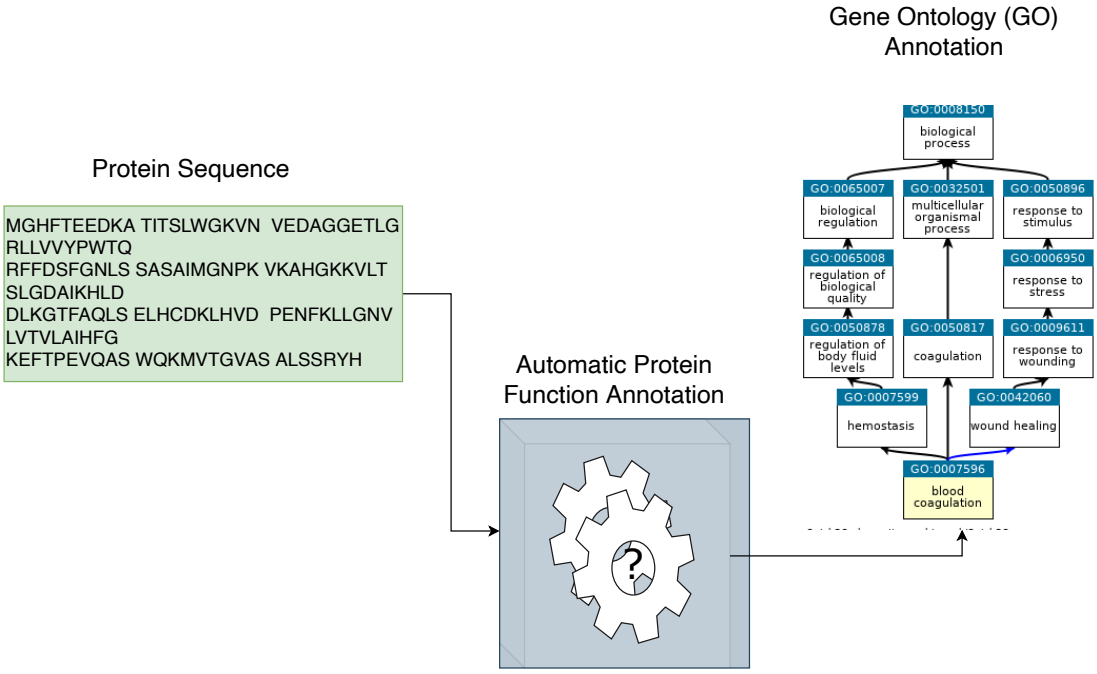


FIGURE 2.9 – Protein function annotation by computational approaches.

integrate the protein entry along with all of the annotations gathered from the previous steps by biocurators.

Automatic Protein Function Annotation

Manual annotation is laborious, time consuming, and expensive. It is not possible to keep pace with the exponential growth of protein sequences in the public databases. Therefore, a general interest in the community is to build computational tools for automatic protein function annotation to accelerate the annotation process and increase the annotated proteins in the database. The central objective of automatic protein function annotation is to assign annotations from EC or GO directly using computer algorithms leveraging the existing annotations, literature, and expert knowledge. The automatic annotation procedure envision to replace the labor of biocurators by placing complex computational techniques in the pipeline (Figure 2.9)

There are a number of computational approaches proposed for automatic protein function annotation using different complex algorithms, for example, mining association rules, machine learning, support vector machine, neural networks, deep learning etc.

2.3 State-of-the-art Research

In the following subsection, we will discuss various computational techniques that have been developed to annotate protein automatically. The techniques are discussed categorically based on the types of computational approaches they adopt to find the best annotations for the query proteins.

2.3.1 Annotation by Association Rules

Guilt-by-association or annotation by association rules is still the dominant approach for automatic protein function annotation problem. The main idea here is to discover the statistical association of un-annotated proteins with annotated proteins and then transferring the known annotations. That is, when a new protein arrives in the database, it is matched against known proteins and the known annotations are transferred from the best matched proteins. Primarily the UniProtKB annotates the proteins using manually curated expertly defined rules in **UniRule** [Gattiker et al., 2003] that uses a large list of "if-then" rules.

These rules come from expert knowledge of scientists and curators who regularly look into experimental data coming from the scientific journals and read articles for relevant information regarding the protein to be annotated. Once few of the proteins are manually annotated, they serve as the seeds to devise the UniRule rules. These rules are tested by experienced curators. UniRule rules can annotate protein properties such as the protein name, function, catalytic activity, pathway membership, and sub-cellular location, along with sequence specific information, such as the positions of post-translational modifications and active sites. In general, UniRule rules are very reliable. However, this is a laborious and time consuming process that very much depends on human expertise. Moreover, the annotation coverage is very low compared to the rate of accumulation of proteins in UniProtKB/TrEMBL.

To alleviate difficulties with curating manual rules in UniRule, a data-driven machine learning technique have been put into the annotation pipeline. The system is called Statistical Automatic Annotation System (**SAAS**) [Kretschmann et al., 2001] and it acts as a complementary system to support the labour-intensive UniRule system by generating automatic annotation rules. SAAS generates automatic annotation rules using decision tree algorithm [Quinlan, 1986] applied over the already annotated entries in UniProtKB/SwissProt.

Very recently, in August, 2020, SAAS was replaced by another updated rule based system called **ARBA** (Association-Rule-Based Annotator) [Saidi et al., 2017, Boudellioua et al., 2016] which is now in the mainstream annotation pipeline in UniProtKB to annotate UniProtKB/TrEMBL protein entries. ARBA is a multi-class machine learning system trained with expertly annotated entries in UniProtKB/SwissProt. It uses association rule mining to identify potential rules that are associated with certain types of function. It generates precise annotation models i.e. rules and performs with relatively high accuracy and coverage.

Along with functions, ARBA rules can annotate other protein properties such as catalytic activity, pathway membership, sub-cellular location and protein names. It generates around 23 thousands models/rules that associates InterPro signatures and taxonomic identities to corresponding functions and the rules are learned completely from the annotated entries of UniProtKB/SwissProt.

There are other lines of research that perform association-based annotation using sequence template or structural template matching. For example, [Dobson and Doig, 2005, Yang et al., 2015, Nagao Chioko and Kenji, 2014] discuss structural-similarity-based approaches to transfer annotations from known/annotated proteins. These works talk about associating proteins with Enzyme Commission (EC) number. In a similar way, there are many approaches based on sequence similarity such as [Rahman et al., 2014, Kumar and Skolnick, 2012, Quester and Schomburg, 2011, Yu et al., 2009].

Blast2GO or B2G [Conesa et al., 2005] is a sequence-similarity-based functional annotation suite for GO annotation. By using BLAST [Mount, 2007] , B2G retrieves all GO annotations for the hit sequences, together with their evidence code which is interpreted as an index of the trustworthiness of the GO annotation. To find specific annotations with certain level of reliability,

an annotation score (AS) is computed for each candidate GO, which is composed of two additive terms. The first direct term (DT), represents the highest hit similarity of this GO term weighted by a factor corresponding to its evidence code. The second term (AT) of the AS provides the possibility of abstraction. This term multiplies the number of total terms unified at the node by a user-defined GO weight factor that controls the possibility and strength of abstraction. Finally, lowest terms per branch that lie over a user-defined threshold are selected. Once GO annotation is available through B2G, the application offers the possibility of direct statistical analysis on gene function information.

A structure-based protein function annotation approach called **COFACTOR** is described in [Roy et al., 2012, Yang et al., 2015]. The updated version of COFACTOR [Zhang et al., 2017] combines information about protein structure and sequence homologs along with Protein-Protein Interaction (PPI) networks to form a hybrid model for jointly predicting GO terms, EC numbers, and ligand-binding. The input of the model is a protein sequence that needs to be annotated with appropriate function. The sequence is then translated into a 3-D structure with the help of in-house 3-D structure prediction tool. After that, a template matching algorithm is run to find the closest peers that have the highest structural similarity to find the homologs. In a similar fashion, the sequence-based homologs are also identified by using BLAST. And interaction partners are also identified from PPI network. All these elements form the basis for annotation transfer from homologous proteins to the query protein.

2.3.2 Annotation By Machine Learning

Apart from direct rule-based annotation techniques like UniRule, SAAS or ARBA, there are many other approaches for automatic protein function annotation using machine learning techniques that explore sequence encoding, functional domain similarity, and structural similarity.

At first, we will discuss the machine learning techniques that have been used to predict the EC number.

Several machine learning methods like k-nearest neighbor (KNN), support vector machines (SVM), artificial Neural Network (ANN), convolutional neural network (CNN), recurrent neural network (RNN) specially, Long-short-term memory (LSTM) have been studied for sequence modeling and function prediction, for example, in [des Jardins et al., 1997, Nagao Chioko and Kenji, 2014, Li et al., 2016, Nasibov and Kandemir-Cavas, 2009, Li et al., 2018, Shen and Chou, 2007, Volpato et al., 2013, Huang et al., 2007, Lu et al., 2007].

EzyPred [Shen and Chou, 2007] is a KNN-based method that adopts a top-down approach for predicting main class and sub-class of EC number. EzyPred works on protein sequences only to perform the annotation task. It starts by predicting whether or not an input protein sequence is an enzyme. Then, EzyPred proceeds by predicting its main EC class and subclass. EzyPred uses pseudo amino acid composition [Chou, 2009] and functional encoding by exploiting functional and evolutionary information of proteins. Based on two features, EzyPred proposes a modified KNN classifier called OET-KNN (Optimized Evidence-Theoretic KNN). Although EzyPred performs well in terms of accuracy, it predicts only the first two digits of a four-digit EC number. Thus, its predictions are not very specific.

A machine learning-based approach called **SVM-Prot** that uses SVM for classification is proposed in [Cai et al., 2003, Cai et al., 2004, Cai and Chou, 2005]. And in 2016 [Li et al., 2016], the performance of SVMProt is improved by adding two more classifiers : 1) KNN , and 2) probabilistic neural networks (PNN). This approach uses important physico-chemical properties such as molecular weight, polarity, hydrophobicity, surface tension, charge, normalized van der Waals volume, polarizability, secondary structure, solvent accessibility, solubility, and the num-

bers of hydrogen bond donors and acceptors in side chain atoms to transform protein sequences into numerical feature representations. A web service is launched and made public to perform experiment using SVMProt.

EFICAz [Tian et al., 2004, Arakaki et al., 2009, Kumar and Skolnick, 2012] presents a method for Enzyme Function Inference by combined approach. EFICAz combines predictions from four different methods using ; (i) functionally discriminating residues (FDRs) in enzyme families obtained by the authors' "CHIEFc" procedure (Conservation-controlled HMM iterative procedure for enzyme family classification), (ii) pairwise sequence comparison using a family-specific sequence identity threshold, (iii) FDRs in multiple Pfam enzyme families, and (iv) recognition of multiple Prosite patterns of high specificity.

A deep-learning approach called **DEEPre** [Li et al., 2018] predicts EC numbers putting together multiple tools and techniques including PSI-Blast [Altschul et al., 1997], HMMER [Finn et al., 2011], CNN, RNN, and sequence encoding using position specific scoring matrix (PSSM) to perform dimensionality uniformization, feature selection, and classification model training. In recent years, deep learning has been applied in many computational biology and healthcare prediction tasks and achieved state-of-the-art performance. However, deep learning approaches can suffer from interpretability issues which is crucial in medical research and clinical decision-making [Che et al., 2017].

In **ECPred** [Dalkiran et al., 2018], the authors describe a hierarchical prediction model. The model starts by predicting if a query sequence is an enzyme or non-enzyme. Once the query sequence is found to be an enzyme, ECPred predicts the main class to which the query sequence belongs. In the similar fashion, it follows the hierarchy of the EC numbering system to find the sub-class, sub-sub-class and sub-sub-sub-class. ECPred learns independent classifiers for 858 EC classes including 6 main classes, 55 subclass classes, 163 sub-subclass classes, and 634 sub-sub-sub classes. The independent predictors that make up ECPred are SPMap, BLAST-kNN and Pepstats-SVM which are based on sub-sequences, sequence similarities, and the physico-chemical features of amino acids, respectively.

At this point, we will extend the discussion into protein function prediction using GO terms. We provide a brief overview of few of the state-of-the-art GO prediction tools that propose ensemble approaches and exploit different feature engineering such as sequence encoding, functional domain similarity and structural similarity, protein interaction network etc.

PANNZER [Medlar et al., 2018, Koskinen et al., 2015] uses weighted KNN approach with statistical testing to predict protein functional annotation. It starts with a sequence search against sequence database, to obtain a Sequence Similarity Result List (SSRL). To avoid biases towards large sequence families due to locally similar but globally dissimilar sequences, there is a limitation on the number of sequences taken for analysis. Focus is only on the sequences that obtained strongest results from sequence scoring and hence the authors apply pre-set filtering thresholds on alignment coverage, identity percentage, sequence length and informative descriptions. Non-linear weighting of taxonomic distances is another source of information used in PANNZER, corrected with a non-linear similarity function between the descriptions of compared query and target sequence. The second step of the PANNZER pipeline is to re-score the sequence hits using a sparse regression model that combines various signals from sequence alignment and non-linear taxonomic distance score and the weighted sum of score functions obtained is optimized against weighted similarity. In the final regression model all terms that had negative correlation with predicted variable from the model are excluded and final score is obtained.

GoFDR [Gong et al., 2016] is a sequence-alignment-based algorithm that runs BLAST [Mount, 2007] or PSI-BLAST [Altschul et al., 1997], for a query protein, to obtain multiple sequence alignment (MSA) over the query sequence. It then identifies all GO terms associated with the sequences in

MSA, and determines the functionally discriminating residues (FDRs) for each GO term. These FDRs are used to generate a position-specific scoring matrix (PSSM) which is then used to compute the score between the query protein and a GO term, followed by a raw score adjustment step to convert the raw score into a probability.

DeepGO [Kulmanov et al., 2017] uses deep learning to learn low rank latent features from protein sequences as well as from a cross-species protein–protein interaction (PPI) network. It utilizes the dependencies between GO classes as background information to construct a deep learning model. Input to the model is amino acid (AA) sequence of proteins in the form of 3-mers, composed of three consecutive AAs, which is represented as one-hot encoding vectors followed by a dense embedding layer. A 1D convolution is applied over protein sequence data and redundant information from the resulting feature map is discarded through temporal max-pooling. In addition, DeepGO uses PPI networks of multiple species, to generate knowledge graph embedding, which are with output of the max-pooling layer to form a combined feature vector. Finally, fully connected layers for each class in GO are used to create a hierarchical classification neural network model that encodes transitivity of subclass relations. The main advantage of this approach is that it does not rely on manually crafted features and is therefore an entirely data-driven approach.

2.3.3 Annotation Using Natural Language Processing Tools

There is another interesting line of research that explores the recent advancement in the field of natural language processing. One of the important tasks in natural language processing is to classify texts into classes such as tags, categories, labels, and so on. Text classification is widely used in web search, information retrieval, ranking and document classification. Due to recent successes, neural-network-based models are prevalent in text classifications. Learning distributed representations of words (also known as word embedding) in a vector space facilitates in achieving better performance in downstream natural language processing tasks by grouping similar words together [Mikolov et al., 2013]. Word embedding includes language modeling and feature learning techniques where words or phrases from the vocabulary are mapped to vectors of real numbers. Mathematically, it involves embedding word from high dimensional space to a continuous low dimensional vector space. Word embedding has potential to be used in learning vector representation of proteins based on their amino sequences. However, protein sequences and natural texts are fundamentally different, even though both of them are strings of characters.

Natural language texts possess a defined linguistic structure containing an array of words delimited by various punctuation marks. Whereas biological texts such as protein sequences are strings of letters selected from an alphabet consisting of 20 letters, each representing an amino acid [Kimothi et al., 2016]. Essentially, one string stands for a single protein. Unlike natural texts, there is no way of formally defining words or phrases in protein sequences. Therefore, using a text classification model requires further pre-processing of protein sequences. The most common way of pre-processing is to break the sequences into biological words commonly known as K-mers that are smaller units of size k composed of k consecutive letters from the alphabet. The pre-processing can be done in two different ways.

1) Overlapping k-mers achieved by moving a k-size window over the sequence. That means, given a protein sequence, a predefined window of size k is moved from the beginning to the end. At each movement, the window is moved by one letter. The k number of characters inside the window forms the word. This way, the window is moved till the end of the protein sequence is reached. All of the words collected in this process are placed in order to form the sentence. To explain the process, let us consider the following example. Let us break an imaginary short sequence

"$MAPPSVFSEV$" into overlapping 3-mers. The window size is 3 and it is moved from the beginning to the end. The corresponding 3-mers that built the sentence are MAP, APP, PPS, PSV, SVF, VFS, FSE, and SEV. Therefore, the biological sequence "$MAPPSVFSEV$" is transformed into following space delimited sentence : *MAP APP PPS PSV SVF VFS, FSE, SEV* ;

(2) Using non-overlapping k-mers, k sequences of k-mers are generated by splitting the original AA sequence into non-overlapping words of k consecutive letters with a starting position moved by one letter for each newly generated sequence [Asgari and Mofrad, 2015, Kimothi et al., 2016]. Like overlapping k-mers, a window of size k is moved over the sequence. However, unlike previous approach, non-overlapping window is moved by k characters at each movement so that there is no overlap between two consecutive k-mers. This process is run for k times to produce k number of sentences against a single protein sequence. At each time, the window begins from a letter ahead of the previous iteration. For example, for the sequence "$MAPPSVFSEV$", considering 3-mers, the 3 newly generated space delimited sequences are as follows :

1. *MAP PSV FSE* : The window starts at first letter from the beginning
2. *APP SVF SEV* : The window starts at second letter from the beginning
3. *PPS VFS* : The window begins at third letter from the beginning

Non-overlapping k-mers have been used for learning word embedding tasks and have been shown to have better prediction accuracy when applied to protein family classification task [Asgari and Mofrad, 2015]. Overlapping k-mers are widely used in homology-based sequence search from large databases of protein sequences as in the case of [Altschul et al., 1997]. The works in [Kimothi et al., 2016, Asgari and Mofrad, 2015, Matsuda et al., 2005] present unsupervised word embedding-based protein classification techniques using continuous bag of words (CBOW) and Skip-gram model proposed by [Mikolov et al., 2013]. In [Asgari and Mofrad, 2015], authors explore non-overlapping 3-mer embeddings and apply the method for protein family classification. Although, they show an improved performance for protein domain classification, they have not explored the problem of functional annotation of proteins using EC number or GO terms.

2.3.4 Network-Based Protein Function Annotation

Recently, the notion of network science has attracted great attention across many scientific communities [Barabási, 2003]. Network science has become a multidisciplinary area of research due to its ability to describe complex intertwined systems. It has found applications in many real-world scenarios from banking and the internet routing to modeling the human brain and understanding complex biological process. Several approaches for annotating protein function have used network science, particularly neighborhood-based techniques for protein-to-protein propagation of functional information using protein-protein interaction (PPI) networks and Gene Ontology terms [Schwikowski et al., 2000, Zhao et al., 2016, Hishigaki et al., 2001, Chua et al., 2006, Nabieva et al., 2005]. The general belief is that the interacting proteins share similar functional behaviours. A particular feature of biological networks is that they often require expert biological knowledge to fully understand and exploit the network.

Graph representation learning also known as network embedding/graph embedding has become a successful approach in network data analysis. The objective of the network embedding models is to optimize structural similarities or distances between the nodes to encode it as similarities or distances in a low-rank embedding space [Nelson et al., 2019]. Network embedding has shown promising results in performing function annotation task carried in various context. For

example, HANDL [Lim et al., 2018] leverages cross-species graph kernels for biologically meaningful network embedding for protein function prediction. OhmNet [Zitnik and Leskovec, 2017] learns network embedding from multi-layer tissue-specific protein network for predicting tissue-specific protein function. One of the limitations of these approach is that they works on homogeneous network i.e. the nodes are of same types. However, biological entities interact with different types of entities that play crucial roles in understanding protein function annotation. Knowledge graph provides a powerful avenue for representing network of heterogeneous biological entities [Mohamed et al., 2020, Nicholson and Greene, 2020] connected with a multitude of relations.

Knowledge Graphs

Large knowledge graphs are increasingly adding value to various applications that require machines to recognize and understand queries and their semantics, as in search or question answering systems [Krompaß et al., 2015]. A knowledge graph is a multi-relational graph composed of nodes representing various types of entities, and edges representing various types of relations [Wang et al., 2017b]. Knowledge graph is a powerful approach to integrate data coming from various sources. Knowledge graphs are becoming a robust way of representing relational data for applications in various industrial as well as academic sectors such as biological systems [Dumontier et al., 2014], lexical information [Miller, 1995], semantic search engines [Qian, 2013], question answering systems [Ferrucci et al., 2010] and general knowledge repositories [Mitchell et al., 2018b].

Biological networks are ubiquitous. They consist of interconnected entities that function together to form complex biological systems. Deciphering different biological process as well as understanding diseases and drug discovery requires to understand the interactions of these biological entities. Knowledge graphs provide an feasible, automated and machine comprehensible means to model interconnected multi-relational entities from diverse biological data [Mohamed et al., 2020]. Knowledge graphs supports a wide range of biomedical applications by finding new treatments for existing drugs [Himmelstein et al., 2017], aiding efforts to diagnose patients [Choi et al., 2017] and identifying associations between diseases and biomolecules [Shen et al., 2017].

Due to the availability of biomedical data and many biomedical databases, it has become now feasible to build biomedical knowledge graphs and perform reasoning over such knowledge graphs to infer important insights and drive new discoveries. Possibilities have also been opened in the domain of protein function annotation by building knowledge graphs with proteins and associated information from annotated proteins. An inference mechanism can be devised to perform annotation task for new proteins. Knowledge graphs, in general, are incomplete. There remain many missing links that demand to be discovered. Discovering new links in a knowledge graph mainly rely on knowledge graph embedding techniques. And it leverages the recent progress in the field of machine learning especially deep learning-based graph embedding. Biomedical knowledge graphs structurally represent the interrelations among biological concepts as nodes and edges of the graph [Nicholson and Greene, 2020]. A comprehensive review on : 1) constructing biomedical knowledge graphs, 2) biomedical contexts and challenges, and 3) applying representation learning on knowledge graphs for biomedical discoveries can be found in [Nicholson and Greene, 2020].

Knowledge Graph Embedding

In the recent years, research in knowledge graph has found a new surge in the machine learning community, specially, in the domain of representation learning. Representation learning is a branch of machine learning where the objective is to learn latent features from data using

unsupervised techniques [Goodfellow et al., 2016]. In other words, representation learning uses unsupervised machine learning, especially, deep learning models to learn low rank vectors for data points. Learning representation on knowledge graph i.e. knowledge graph embedding aims at transforming entities and relations of knowledge graph into numerical vectors by encoding its structural properties. In general, knowledge graph embedding (KGE) models learn low-rank vector representations for the knowledge graph entities and relations. These numerical vectors can be used to perform downstream predictive tasks, for example, link prediction, node prediction. And thus it helps in effective and scalable discoveries in knowledge graphs [Mohamed et al., 2019]. The advancement in knowledge graph embedding (KGE) research has shown success in a wide range of tasks such as link prediction [Rossi et al., 2020, Bordes et al., 2013], entity resolution [Bordes et al., 2014, Nickel et al., 2011] , and entity classification [Nickel et al., 2012].

The reasoning approaches often utilise the power of representation learning that maps the entities and edges into low dimensional vector space. This mapping of nodes and edges into a low dimensional space encodes the structural aspects of the knowledge graph in the form of vectors of numbers. These vectors, eventually, facilitate in the downstream tasks as they can be readily fed into machine learning models like neural networks, logistic regression. Knowledge graph embedding techniques can be categorized in the following groups :

Distance-based transnational approaches, for example, TransE [Bordes et al., 2013], projects nodes and edge types in the low dimensional vector space following the vector translation operation defined as

$$s + p \approx o$$

, where s is the source node or subject or head, p is the relation type or predicate and o is the target node or object representation. This means that starting from the subject node (s), adding the relation type (p), one arrives at the object node (o). One of the concerns with TransE is that during negative sampling it perturbs a positive triple (s, p, o) and randomly chose an object node (o') from the available entities in the graph to generate a negative example (s, p, o'). This approach has a problem, specially, in the case of protein function annotation. To explain the problem, let us consider the following (Figure 2.10) where a protein u is connected with 3 Interpro domains (d_1, d_2, d_3) with a relation type r_1 that constructs its domain architecture. Also, it is annotated with two GO terms $(g1, g2)$ and hold a relation type r_2. In the context of knowledge graph, the edges can be written as (u, r_1, d_1), (u, r_1, d_2), (u, r_1, d_3), (u, r_2, g_1), (u, r_2, g_2). These five edges are treated as positive facts in the knowledge graph. Therefore, according to definition, this knowledge graph has $V = u, d_1, d_2, d_3, g_1, g_2$ and $R = r_1, r_2$. To construct negative facts from the first edge (u, r_1, d_1), TransE replaces d_1 with random entity from V. In some cases, it may happen that the random sampler picks one of the (d_1, d_2, d_3) and proposes (u, r_1, d_3) as negative edge which in fact a positive edge. In fact, for knowledge graph where a single entity is connected to multiple entities with a single relation type, this type of negative sample can hurt the learning process.

Matrix Factorization techniques depend on decomposing large adjacency matrix into constituent low dimensional matrices representing the low dimensional vectors for the entities or nodes. For example, Rescal [Nickel et al., 2011, Nickel et al., 2012] factorizes an adjacency tensor A of size $|V| \times |V| \times |R|$ into two matrices of size $|V| \times d$ holding entity embedding in d dimensional space and another tensor of size $d \times d \times |R|$, where V is the set of entities, R is the set of edge types. Each edge type is essentially learnt as a matrix of $d \times d$. One of the drawbacks of tensor factorization-based approaches is that depending on the size of the knowledge graph the computational complexity can be very high. Rescal is an efficient embedding technique for relatively small graphs in terms of edge types. As in biomedical applications, the relation types

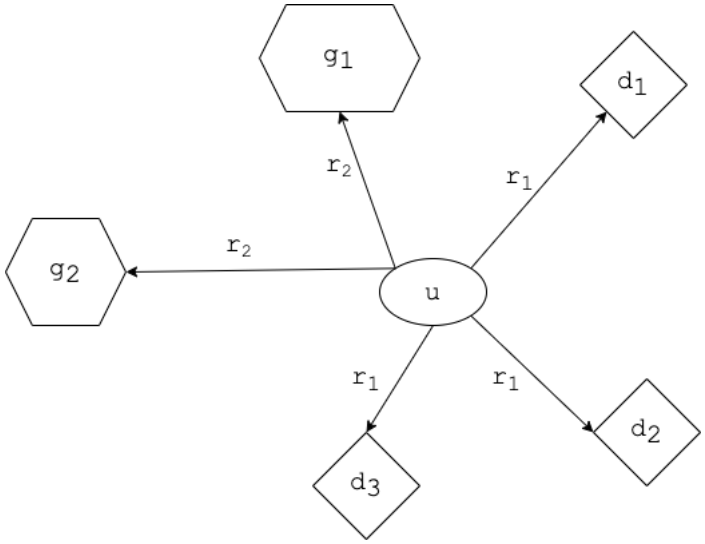


FIGURE 2.10 – Knowledge Graph for toy example

and node types grow pretty fast as the number of data sources increases, the computational complexity might be prohibiting for large scale biomedical graphs.

Neural Network is the *de facto* standard now for modeling knowledge graph embedding following the advancement in graph representation learning. Considering the broad perspective, there are two streams of work that utilize neural network : 1) neural message passing is the widely accepted method for graph representation learning, and 2) Word2vec-based node presentation learning by performing random-walk on the graph. In case of neural message passing technique, the iterative accumulation of information from neighbor-nodes conditioned on relation type is fed into CNN-type neural network that acts as encoder and encodes the graph structure into vectors. For example, [Gysi et al., 2020] explored a biomedical knowledge graph to perform drug repurposing - finding new use cases of existing drugs - for COVID-19 disease. Authors have taken a network medicine approach where they have analyzed a network of proteins, drugs, and diseases. They have built a knowledge graph of proteins, drugs and diseases connected with following edge types : 1) protein-protein interactions, 2) drug-target associations, 3) disease-protein associations, and 4) drug-disease indications. They applied neural message passing to learn embedding for nodes i.e. proteins, drugs and diseases. To compute the embedding they used graph neural network-based encoder to propagate the information across the network. A decoder is designed to optimize the embeddings to reflect the task at hand, drug repurposing in this case. The form of the convolutional operator that is used in the encoder is as follows :

$$h_i^{(k+1)} = \sigma(\sum_r \sum_{j \in N_r^i} a_r^{ij} W_r^{(k)} h_j^{(k)} + a_r^{ij} h_i^{(k)})$$

where $h_i^{(k)} \in R^{d(k)}$ is latent state at the node v_i at the k^{th} layer of the neural network having $d(k)$ dimension of the representation, r is the relation type, $W_r^{(k)}$ is a type specific parameter matrix, a_r is the attention co-efficient for relation type r. The final embedding is collected as z_i=$h_i^{(k)}$. The decoder takes the form of a scoring that scores the likelihood of a triple (v_i, r, v_j). The method then produces the ranked list of target drugs against the covid-19 disease based on disease-drug similarity using the learnt embedding.

There is another line of work that takes advantages of word embedding techniques borrowed

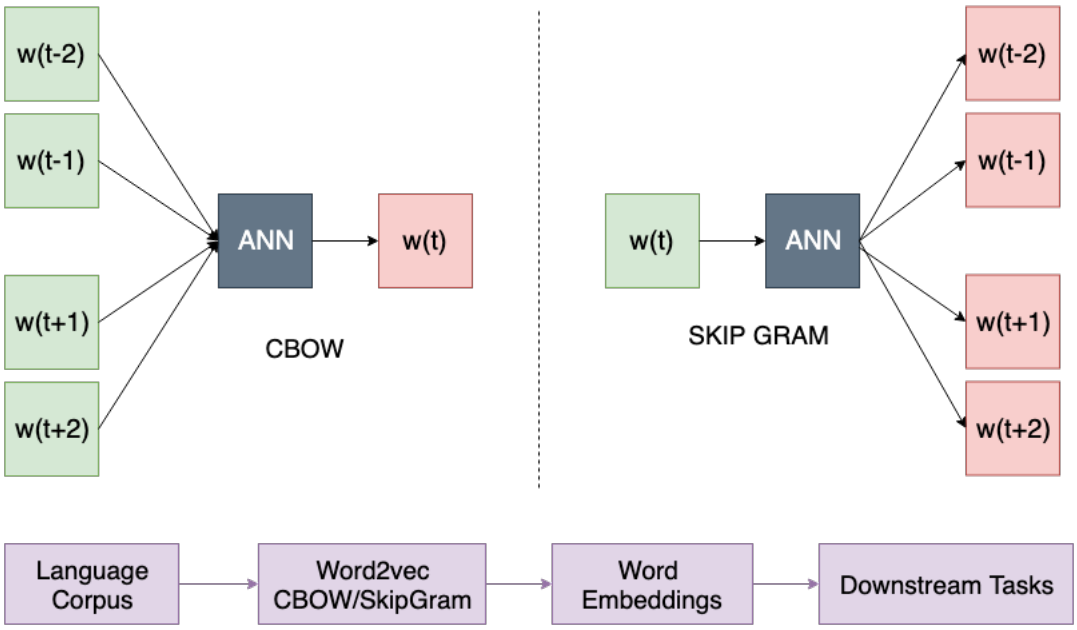


FIGURE 2.11 – CBOW and Skip-gram word embedding model

from natural language processing. In this scenario, the graph is transformed into a corpus of sentences that represent the sequence of nodes gathered using random-walk-based graph traversal. Word2vec [Mikolov et al., 2013] uses skip-gram or Continous-Bag-of-Word (CBOW) model for the purpose of word representation. It is an effective method in natural language processing for finding semantically similar words that helps in fast information retrieval. In Figure 2.11, CBOW and Skip-gram architecture are shown side-by-side. In case of skip-gram, a neural network is trained to predict the context words - words in the vicinity of the fixed window size - given an input word. Shown in the figure, given an input word $w(t)$ in a position t, the context words $w(t-2), w(t-1), w(t+1), w(t+2)$ are predicted. And in the case of CBOW the scenario is just the opposite. Given a set of context words $w(t-2), w(t-1), w(t+1), w(t+2)$, the target of the neural network training is to predict the target word $w(t)$. In each case, the neural network optimizes the word vectors to fit the text corpus. Following the analogy, node embeddings are learnt using word2vec model similar to word embedding. Node2vec [Grover and Leskovec, 2016], DeepWalk [Perozzi et al., 2014] apply word2vec to devise a node representation learning model. In Node2vec, for each node, fixed-length random walk is performed for a set-number of times. These random walks form the corpus of nodes. After that, a skip-gram model is trained from the node corpus in a unsupervised fashion. DeepWalk also follows a similar principle.

Following the success of node2vec, similar approaches have been tried in the case of knowledge graph embedding. RDF2vec [Ristoski and Paulheim, 2016] is such a technique that learn representation of knowledge graph expressed in RDF (Resource description framework) format. RDF2vec uses word2vec model for unsupervised feature extraction from sequences of nodes. In RDF2vec, local information from graph sub-structures computed from graph walks is leveraged to generate sequences of entities for learning latent numerical representations of entities in RDF graphs. Following Figure 2.12 shows a schematic diagram of how RDF2vec works.

Generative adversarial Network (GAN) [Goodfellow et al., 2014] has been a big success in computer vision in generating realistic images after training with sufficient amount of data. The following Figure 2.13 shows the typical schematic diagram of a GAN.

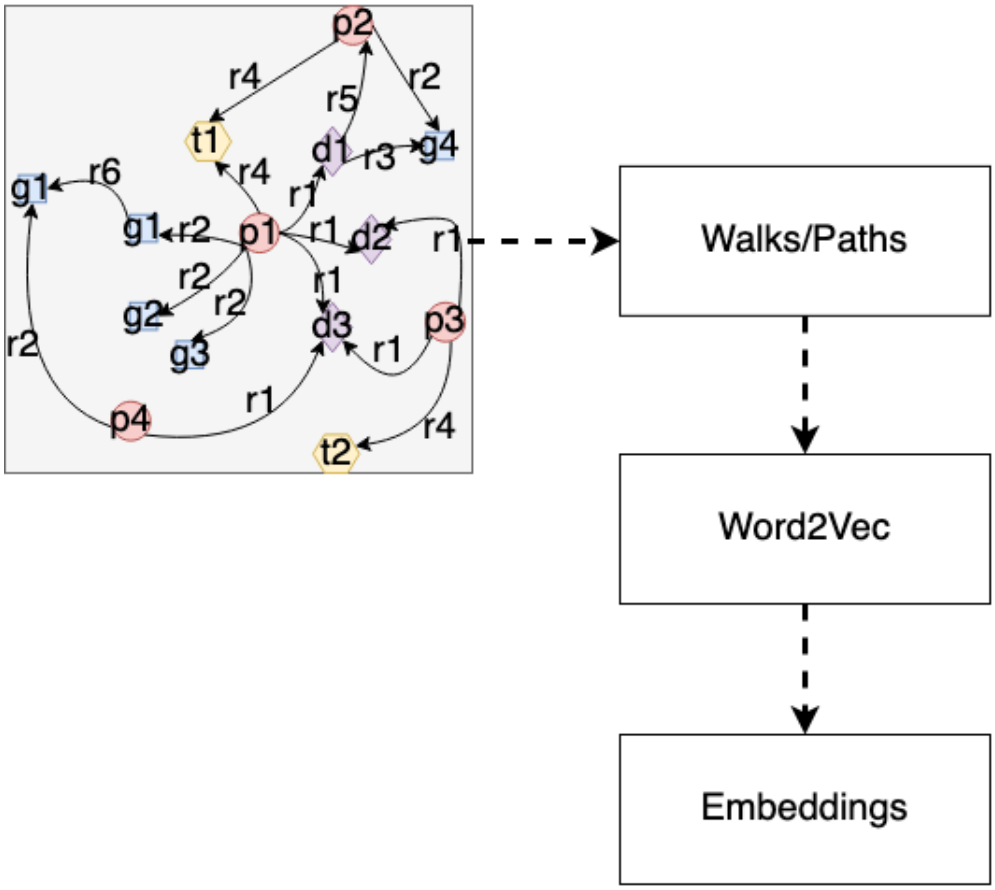


FIGURE 2.12 – RDF2vec schematic diagram

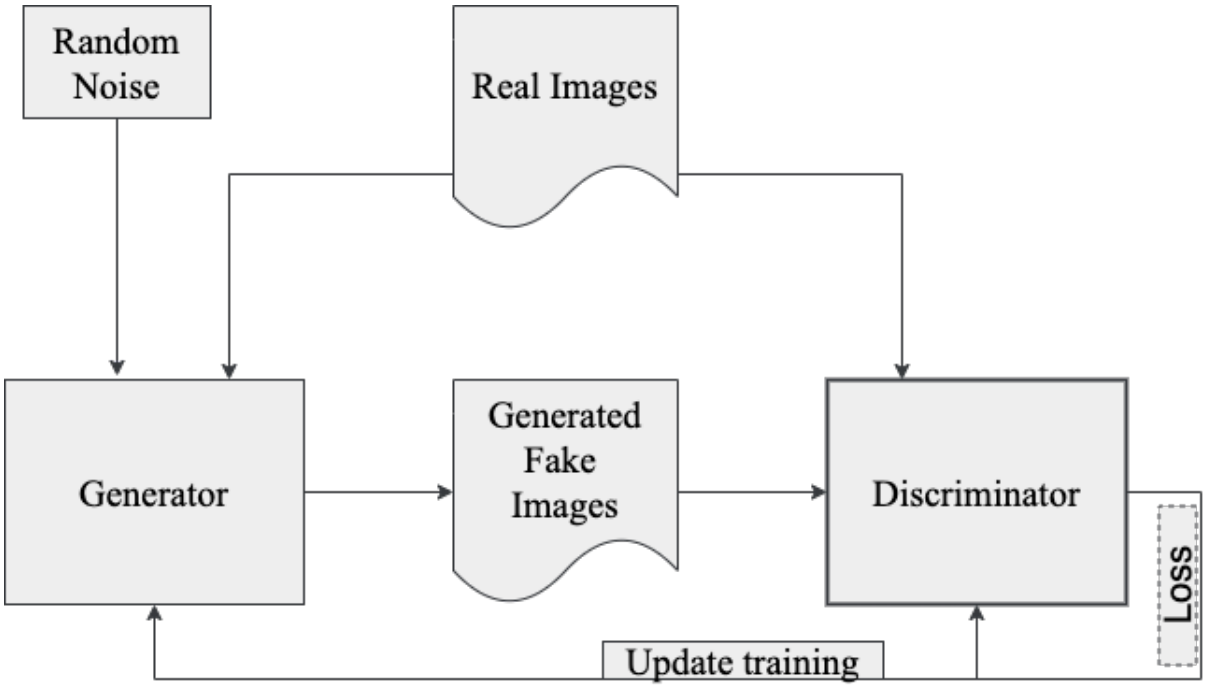


FIGURE 2.13 – Schematic diagram of GAN for computer vision

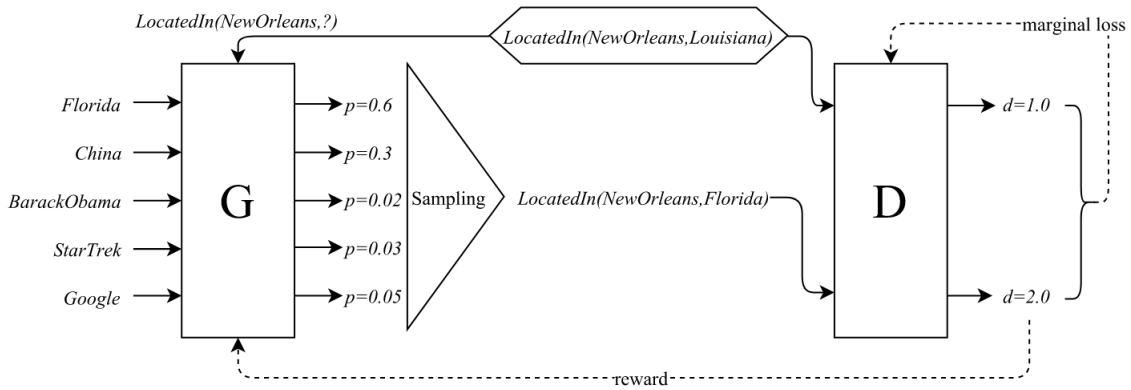


FIGURE 2.14 – KBGAN [Cai and Wang, 2018] for negative sample generation using GAN on knowledge graphs

The main idea of GAN is to perform an adversarial two-player game between a generator and a discriminator. The objective of the discriminator is to maximize the distance between real image and generator image proving generator image is fake. On the contrary, the objective of generator is to generate images that looks like real, thus minimizing the distance between real and generated image. This minimax game continues to achieve equilibrium so that the discriminator has become an expert in identifying fake images and the generator is expert in producing highly realistic fake images.

Very recently, GAN has been repurposed in the context of homogeneous graphs embedding [Wang et al., 2019, Ding et al., 2018]. GraphGAN [Wang et al., 2019] is a GAN-inspired node embedding technique where a GAN like adversarial training is performed to train 1) a discriminator which is a simple cosine similarity function over two nodes (v_i, v_j), and 2) a generator which probabilistically selects node-neighbors from the graph, given a node and following a breadth-first-search (BFS). The same procedure is used to generate negative samples for the discriminator. The discriminator tries to classify the generated nodes as fake nodes whereas the generator optimizes itself by getting reward from the discriminator over its choices of neighbor nodes. During the selection process, the generator starts from a root node, moves a step forward by selecting next node from all neighbors using a softmax probability score and stops moving once it finds the next node rewind back to be the previous node. This way, it selects pairs of nodes that are fed into the discriminator to get reward about its choices. One of the drawbacks of this technique is that it built a entire BFS tree from the graph. This can get computationally intractable when the graph grows to be larger.

On the contrary, GraphSGAN [Ding et al., 2018] follows an approach that sounds more like a GAN. They used neural network architecture to learn the probability distribution. However, both GraphGAN and GraphSGAN work for homogeneous graphs. These techniques do not readily apply in the settings of knowledge graph. Other GAN-like techniques have been adapted for knowledge graph embeddings. For example, IRGAN [Wang et al., 2017a], KBGAN [Cai and Wang, 2018] uses adversarial training to generate negative samples from the fact base or knowledge graph. And then, existing knowledge graph embedding techniques are applied to learn the entity representation. For example, Figure 2.14 is the schematic diagram of how the KBGAN technique uses GAN with policy gradient for sampling negative facts.

Although GAN-based knowledge graph embedding techniques have created a lot of attentions, knowledge graph has not been explored in the context of automatic protein function annotation.

Automatic protein function annotation problem demands for finding right annotations for query proteins. That is, if a query protein is given and the relation is given, we are interested to know which are the functions most likely to be associated with the protein. The problem can be formulated as link prediction task where we want to score the association between a protein and a function conditioned on relation type. This can be achieved by using the embedding and an appropriate similarity function like cosine similarity.

On the other hand, the problem can be seen from the perspective of generating the annotations. In this case, given a protein and an edge type, we are interested to find GO terms that are most likely to be associated with the proteins. Applying domain knowledge and following the work of GraphGAN [Wang et al., 2019] and IRGAN [Wang et al., 2017a], instead of generating negative sample, the model can be trained to select the best annotations for the protein.

As part of this thesis, we build a knowledge graph putting the real world constraints applicable in the case of protein function annotation. We propose a knowledge embedding technique to utilize protein meta-data in function prediction. We follow the works of GraphGAN and IRGAN and devise a discriminator that takes into account negative sample produced by applying the domain-specific knowledge and a generator as a target specific depth-limited random walk to generate annotations. We formulate automatic protein function annotation task as a link prediction problem over a knowledge graph.

Deuxième partie

Protein Function Annotation From Domain Similarity Graphs

3

Graph-Based Protein Function Annotation From Domain Similarity Graphs

In this chapter, we present a complete description of a novel graph-based annotation approach called GrAPFI : Graph based Automatic Protein Function Inference, and also, we present a detailed experimental analysis on six popular reference proteomes from UniProtKB/SwissProt database. GrAPFI is a neighborhood-based classification technique. GrAPFI builds network of proteins using domain and family information, and performs neighborhood-based label propagation for function annotation. GrAPFI explores the functional domain architectures extracted from protein sequence instead of protein secondary structure and direct sequence homology. GrAPFI performs label propagation over weighted protein network to select the best EC annotation.

We compare the performance of GrAPFI with the recently published ECPred approach. Along with ECPred, we also present the accuracy for DEEPre and SVMProt as representative examples of other state-of-the-art EC number prediction approaches. Our analysis shows that GrAPFI gives better annotation performance than these earlier approaches.

3.1 Overview of GrAPFI

GrAPFI is a neighborhood-based classification technique. GrAPFI combines the notion of protein domain similarity with a graph neighborhood inference technique for automatic EC number annotation. More specifically, the functional annotations of reviewed proteins in SwissProt are used to predict those of non-reviewed proteins in TrEMBL using label propagation on a complex network representation of protein sequence data. The GrAPFI algorithm first constructs an undirected weighted graph of the proteins using the domain composition of the reviewed proteins. Then, given a non-reviewed protein, it is integrated in the network on the basis of its own domain composition, and a label propagation algorithm is applied to the protein graph in order to infer appropriate annotations.

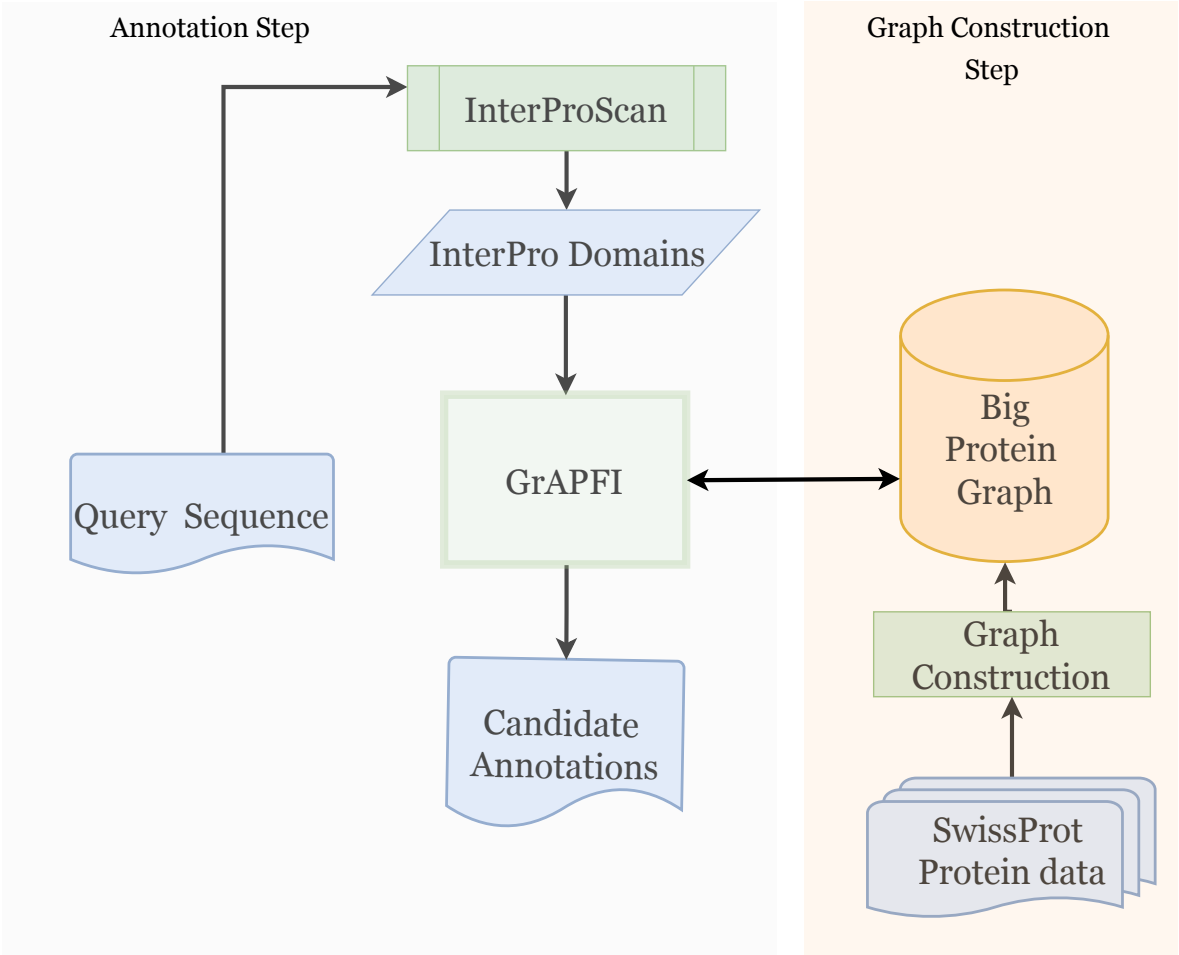


FIGURE 3.1 – The annotation workflow used in GrAPFI. The right-hand portion of the workflow depicts the graph construction using reviewed proteins from the UniprotKB/Swissprot. The left part shows the annotation flow.

3.1.1 Graph construction

We present here a novel way of connecting proteins using their associated InterPro domains and family information. Domains may be considered as natural building blocks of proteins. Due to evolution, protein domains may have gone through changes such as duplication, fusion, recombination to produce proteins with distinct structures and functions [Kummerfeld and Teichmann, 2009]. GrAPFI explores the functional domain architectures extracted from protein sequence instead of protein secondary structure and direct sequence homology. Here, each node of the graph represents a protein, while a link between two nodes means that the proteins exhibit a given level of domain similarity. Thus, each node u is identified by a set of labels $L(u)$, has a set of neighbors $N(u)$, and for every neighbor v it has an associated weight $W_{u,v}$. The overall aim is to propagate labels (i.e. annotations) from nodes having reviewed annotations to similar nodes that lack annotations.

To illustrate the construction of the protein graph, let us consider five proteins with symbolic names $P1$, $P2$, $P3$, $P4$, and $P5$. Let us assume that these proteins are composed of sets of domains $D1 = (d1, d2, d3, d4)$, $D2 = (d1, d3, d5)$, $D3 = (d1, d2, d10)$, $D4 = (d5, d6, d1)$, and

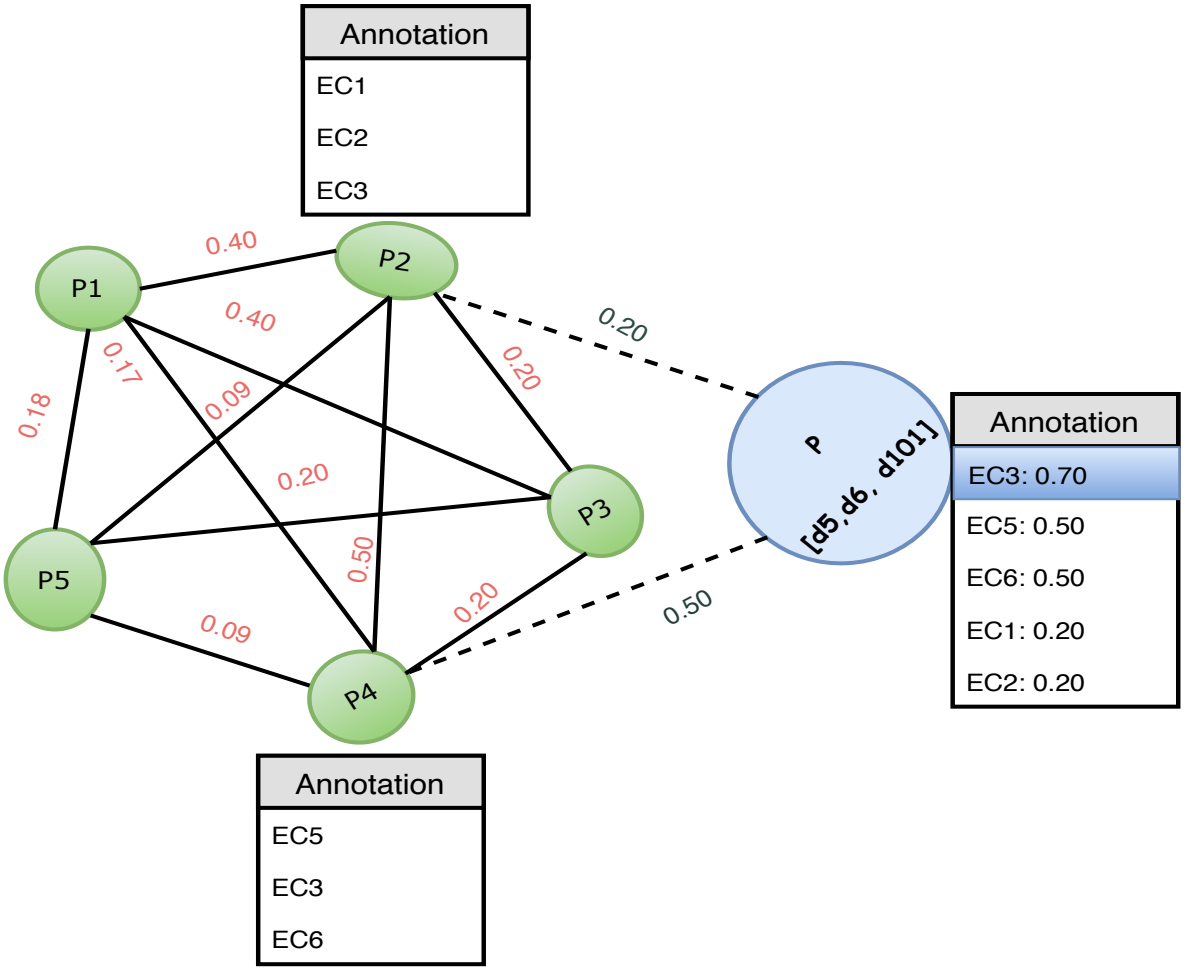


FIGURE 3.2 – Example of EC annotation using label propagation.

$D5 = (d4, d1, d10, d40, d7, d9, d12, d52, d100)$, respectively.

Domain composition of a protein is the set of domains identified in a protein sequence and considered irrespective of order of appearance in the sequence. For example the domain information in $D1 = (d1, d2, d3, d4)$ can be used in any other order $D1 = (d1, d4, d3, d2)$. Therefore, the composition is not strictly linear. The overlapping of domains are not considered as long as the overlapped domains has a new identification number.

It is then evident that proteins $P1$ and $P2$ contain two domains $d1$ and $d3$ in common. Therefore, proteins $P1$ and $P2$ may be linked and the number of shared domains may serve as a link weight given by

$$W_{P1,P2} = |(d1, d2, d3, d4) \cap (d1, d3, d5)| = |(d1, d3)| = 2.$$

In a similar way, proteins $P1$ and $P5$ may be linked with a link weight of $|(d1, d2, d3, d4) \cap (d4, d1, d10, d40, d7, d9, d12, d52, d100)| = |(d1, d4)| = 2$. In both cases, the link weight is 2. However, the link weight computed in this way does not reflect the relative strength of the relationship among the proteins. More specifically, in the first case the two proteins have $|(d1, d2, d3, d4) \cup (d1, d3, d5)| = |(d1, d2, d3, d4, d5)| = 5$ different domains, of which two are shared. In the second case, there are $|(d1, d2, d3, d4) \cup (d4, d1, d10, d40, d7, d9, d12, d52, d100)| = 11$ different domains

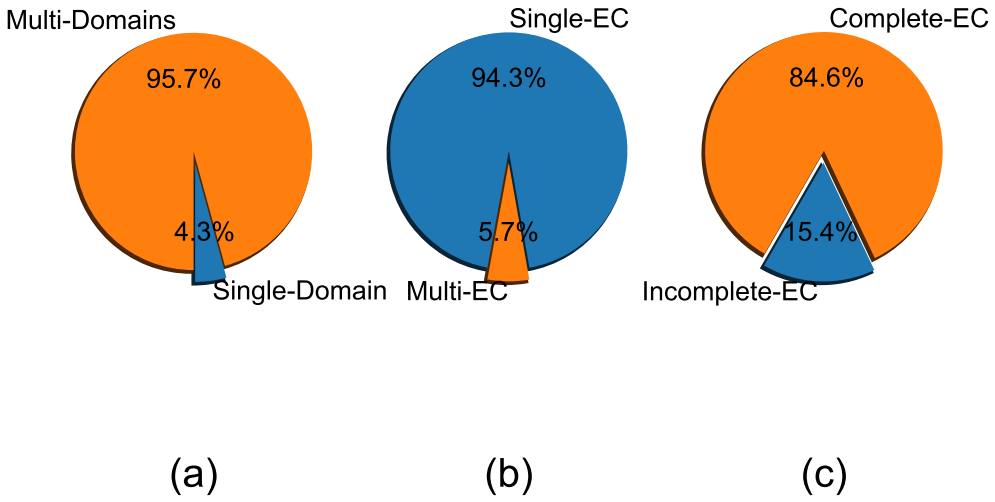


FIGURE 3.3 – Training set statistics like a) proportion of single-domain proteins, b)single-EC proteins and c)proteins with incomplete EC number.

of which two are shared as well. Although two domains are shared in each case, P1 is intuitively more aligned with P2 than P5. Therefore, instead of using the above raw similarity score, we used the Jaccard similarity index, or Jaccard similarity coefficient to better reflect the similarity in composition. This is calculated as $\frac{|A \cap B|}{|A \cup B|}$, where A and B are the two sets of constituent domains. Using the Jaccard similarity index, the link weight for P1 and P2 is calculated as

$$W_{P1,P2} = \frac{|(d1, d2, d3, d4) \cap (d1, d3, d5)|}{|(d1, d2, d3, d4) \cup (d1, d3, d5)|} = \frac{|(d1, d3)|}{|(d1, d2, d3, d4, d5)|} = \frac{2}{5} = 0.4.$$

Similarly, for P1 and P5, the link weight is calculated as

$$W_{P1,P5} = \frac{|(d1, d2, d3, d4) \cap (d4, d1, d10, d40, d7, d9, d12, d52, d100)|}{|(d1, d2, d3, d4) \cup (d4, d1, d10, d40, d7, d9, d12, d52, d100)|} = \frac{2}{11} = 0.18.$$

Using the Jaccard similarity index, the final graph is built in two simple steps. In the first step, the data files that contain reviewed protein information are parsed to collect the constituent domains of each protein. If the training data contains only sequences, InterProScan [Quevillon et al., 2005, Jones et al., 2014] is used to find the domains associated with each of the protein sequences. Then the graph is built using the domain composition of the proteins.

It is worth mentioning that the order of the domains is not maintained while computing Jaccard similarity index. Domain composition for each protein contains the set of unique InterPro signatures found in the sequence.

3.1.2 Enzyme Commission numbers

Enzymes are usually labelled following the Enzyme Commission (EC) system [Cornish-Bowden, 2014], the widely used numerical enzyme classification scheme. It assigns each enzyme a four digits number. This classification system has a hierarchical structure. The first level consists of the six main

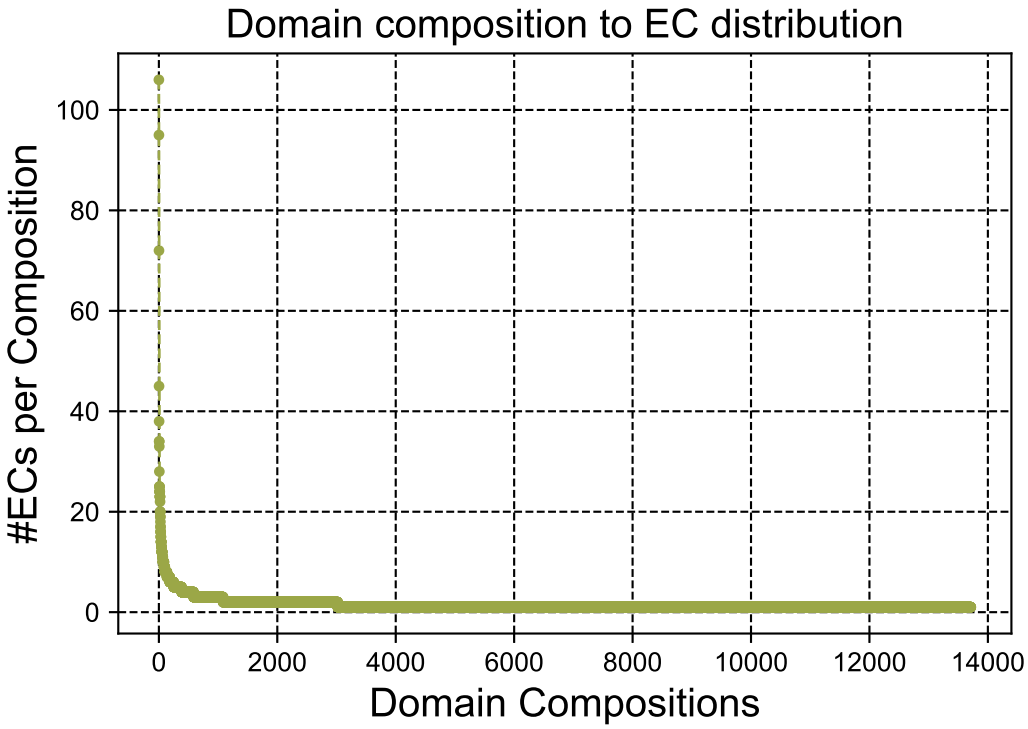


FIGURE 3.4 – Distribution of EC numbers per domain composition

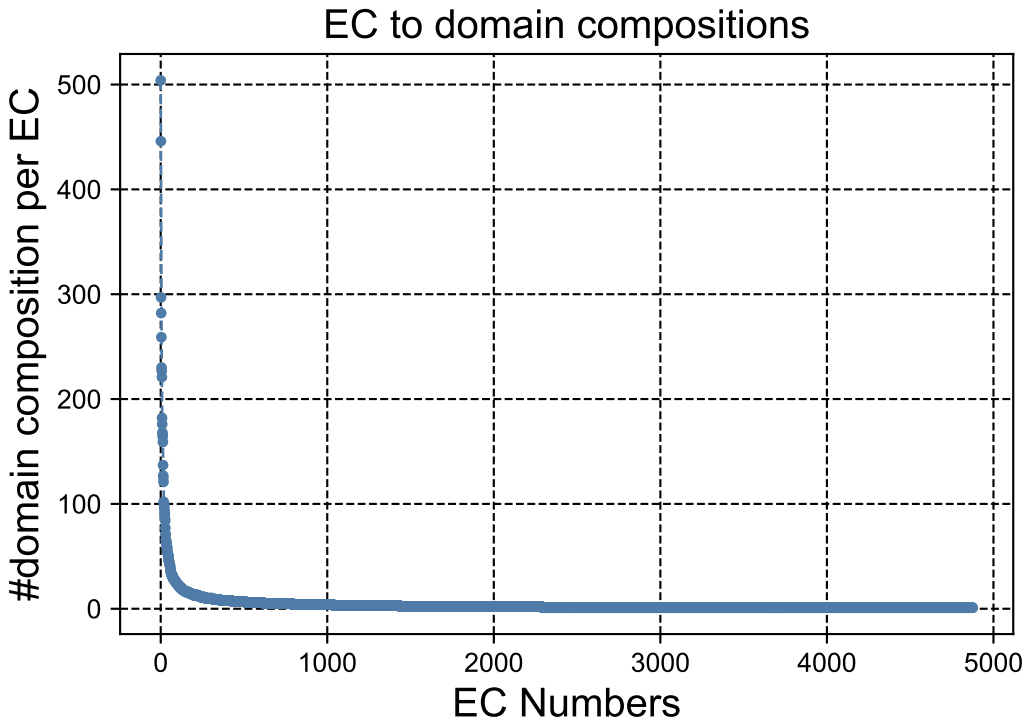


FIGURE 3.5 – Distribution of domain compositions per EC number

enzyme classes : (i) oxidoreductases, (ii) transferases, (iii) hydrolases, (iv) lyases, (v) isomerases and (vi) ligases, represented by the first digit. Each main class node further extends out several subclass nodes, specifying subclasses of the enzymes, represented by the second digit. Similarly, the third digit indicates the sub-subclass and the fourth digit denotes the sub-sub-subclasses.

3.1.3 Label propagation for protein function annotation

After building the graph from the reviewed proteins, the graph is ready to be used for the function annotation of new protein sequences. A neighborhood-based label propagation algorithm is designed to perform the annotation task. Given the constituent domains of an input protein sequence, all of its neighboring proteins and their annotations are retrieved from the graph. Once the neighbors have been obtained, the weighted frequency of the labels are computed using the following formula :

$$f_u^i = \frac{\sum_{v \in N(u)} W_{u,v} \sum_{j \in L(v)} \delta(j,i)}{\sum_{v \in N(u)} W_{u,v}},$$

where f_u^i is the weighted score of the candidate annotation i for the query protein u, and $\delta(j,i)$ is 1 if the function $j \in L(v)$ of the protein v is same as function i, otherwise, 0.

Overall, for a given input sequence, the annotation algorithm works according to the flow diagram shown in Figure 3.1.

The details of the label propagation algorithm is described in Algorithm 1.

Algorithm 1: Label Propagation in a protein graph

Input: A weighted undirected protein graph $G = (V, E)$, Similarity threshold, θ, a query protein u with domain composition D

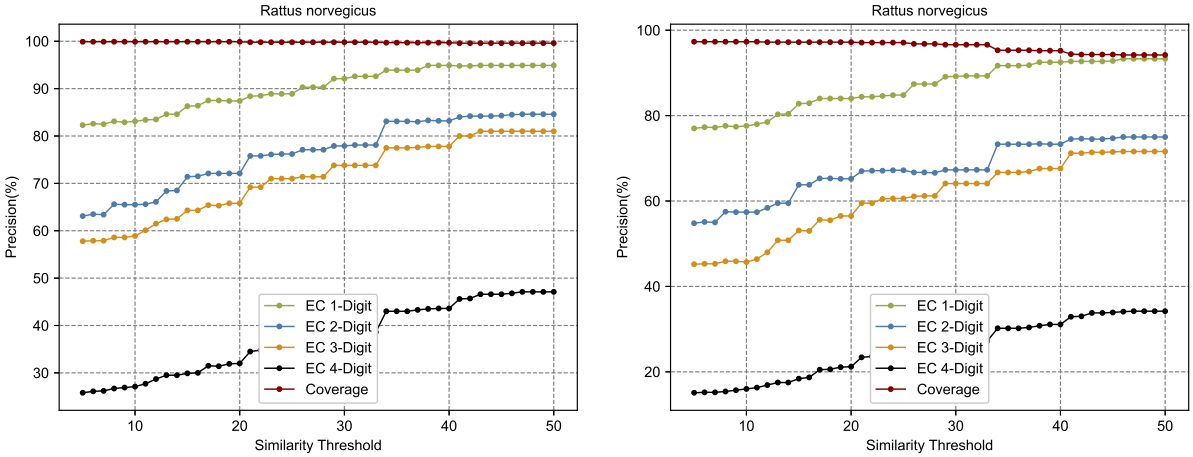
Output: Weighted EC Annotations

```
Annotations ← ∅
N'(u) ← ∅
for each v ∈ N(u) do
    if W_{u,v} >= θ then
        N'(u) ← N'(u) ∪ {v}
ECs ← list of distinct ECs present among the neighbors N'(u)
for i ∈ ECs do
    f_u^i = (Σ_{v∈N'(u)} W_{u,v} Σ_{j∈L(v)} δ(j,i)) / (Σ_{v∈N'(u)} W_{u,v}),
    Annotations ← Annotations ∪ {f_u^i}
Rank the Annotations
Select the top ranked annotations and assign it to the protein u
```

Let us consider a query protein P with a set of domains $D = (d5, d6, d101)$. Our aim is to annotate this protein with an EC Number following the label propagation algorithm, as illustrated in Figure 3.2.

Based on the domain similarity, protein P will have connection with proteins $P2$ and $P4$ in the running example graph. The dotted lines show the links from P to $P2$ and $P4$ in the graph along with the associated weights. Therefore, the protein P will have $P2$ and $P4$ as its neighbors. After finding the neighbors, the functional annotations of all the neighbors are propagated along with the corresponding links to the query proteins weighted by the link weights. All of the functional annotations are ranked based on their cumulative weights. The top-ranked annotation is selected as the best functional annotation for protein P. In this example, the weighted annotations for

P are $EC3$, $EC5$, $EC6$, $EC1$, $EC2$ with cumulative weights of 0.70, 0.50, 0.50, 0.20, and 0.20, respectively. Therefore, the functional annotation for the protein P is $EC3$ as it has the highest weight among the propagated labels. Clearly, it is possible to select more than one high-scoring functional annotation if we wish to propose more than one candidate annotation. Furthermore, node neighbors could be selected in other ways to reflect the requirements of the problem at hand.



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

FIGURE 3.6 – The precision and coverage for different similarity thresholds for *Rattus norvegicus* reference proteome

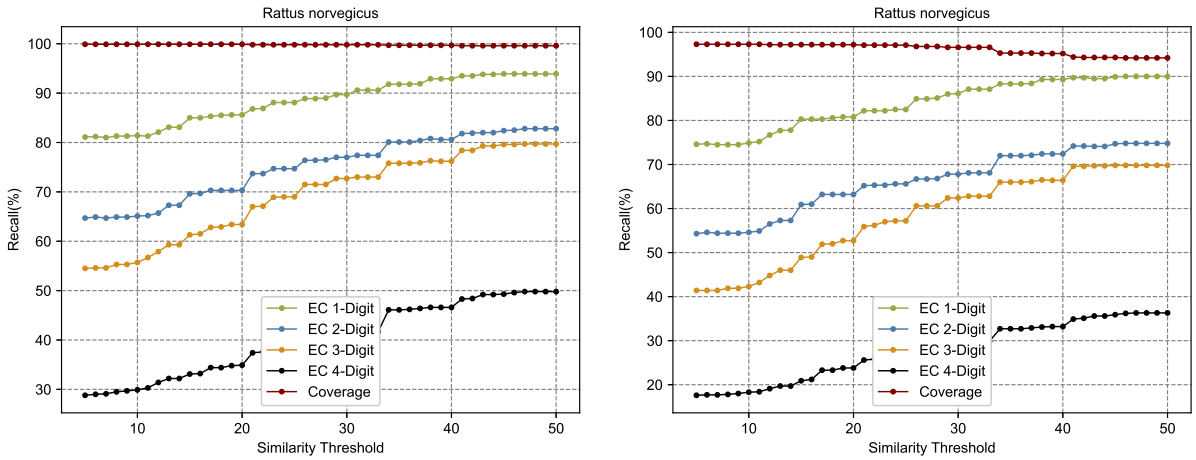
3.1.4 Experiment and Result Analysis

Data Preparation

We have collected 262,564 proteins from the March 2018 release of Uniprot-KB/SwissProt [The UniProt Consortium, 2015] database satisfying the following constraints : (i) each protein must contain at least one InterPro signature and (ii) must be annotated with at least one EC annotation. After getting the protein data from each of the proteins, we have extracted InterPro domain composition and EC annotations. Then, we built the protein network as described in section 3.1.1 Each protein forms its own vertex. We did not preprocess training data to remove redundancy. Rather, while performing annotation, it ignores the same protein if it appears in the neighborhood. For example, for a query protein q, GrAPFI will collect the neighbors satisfying a maximum Jaccard similarity score as well as a minimum similarity score. When the maximum Jaccard similarity is set to less than 1.0, GrAPFI omits the neighbors with exact match in domain composition.

The training network covers 25 level-2, 31 level-3 and 408 level-4 EC classes from 41,618 oxidoreductases, 70,530 transferases, 100,027 hydrolases, 14,677 lyases, 25,551 isomerases, and 29,735 ligases which are linked using 10,866 InterPro signatures.

In the training network, 1) 4.3% of the proteins are single-domain proteins i.e. proteins having only one domain in their domain composition (Figure 3.3(a)), 2) 5.7% of the proteins have more than one EC number assigned with them (Figure 3.3(b)), and 3) Around 15% of the training nodes have incomplete EC annotations i.e. the EC numbers assigned with these proteins do not



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

FIGURE 3.7 – The recall and coverage for different similarity thresholds for *Rattus norvegicus* reference proteome

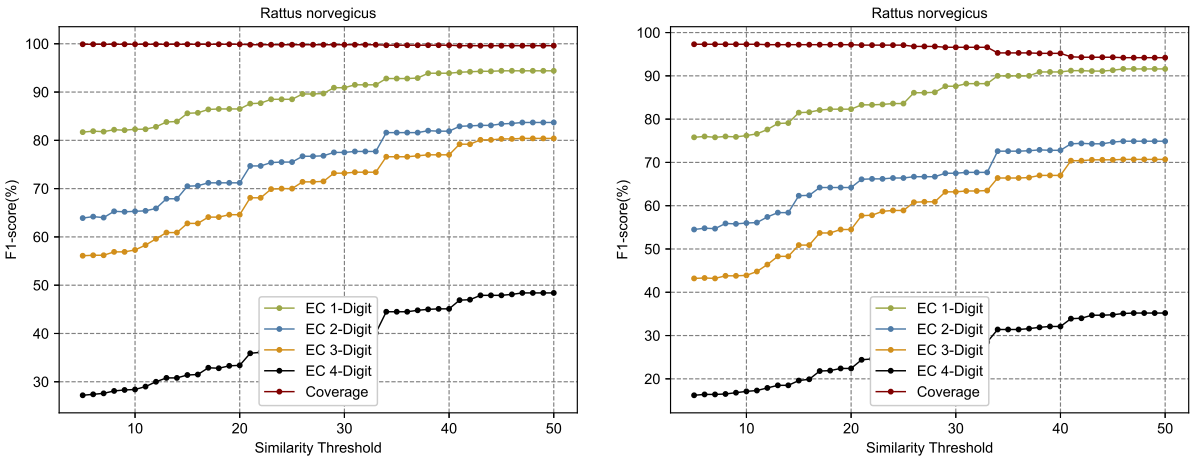
have all four digits. In Figures 3.5 and 3.4, we show the distribution of EC numbers per domain composition and of domain compositions per EC number, respectively. There are 13713 unique domain compositions in the training data. In the X-axis we put the domain compositions and along Y-axis we have the number of different EC annotations found for each domain composition sorted in descending order. It is evident from the figure that few of the domain compositions contain significantly higher number of EC numbers. For example, for some domain composition, there are more than 50 EC numbers found in the training data. We also show the distribution of domain compositions per EC number i.e. the different domain compositions found for each EC annotation shown in Figure 3.5. There are many cases when a higher number of domains compositions are mapped to a single EC. For example, in some cases, it is around 500 distinct domain compositions found against a single EC number. In essence, these two distributions reflect the dominance of many-to-many relationship between domain composition and EC annotation in the training data.

To validate GrAPFI, we used six popular reference proteomes from Uniprot-KB/SwissProt as test cases. The reference proteomes are the following :

1. *Rattus norvegicus* (UP000002494) containing 1,953 proteins,
2. *Mus musculus* (UP000000589) containing 3,682 proteins,
3. *Saccharomyces cerevisiae* (UP000002311) containing 1,581 proteins,
4. *Homo sapiens* (UP000005640) containing 3,843 proteins,
5. *Arabidopsis thaliana* (UP000006548) containing 5,352 proteins, and
6. *E. Coli* (UP000000625) containing 1465 proteins.

For each of the reference proteomes, we collected the InterPro domains and EC labels from Uniprot-KB/Swissprot. We kept only the proteins which have at least one InterPro domain and are annotated with a single EC number.

To prepare the COFACTOR benchmark dataset, we used the 318 protein sequences published in [Zhang et al., 2017], and we ran InterProScan [Jones et al., 2014] on these sequences to get



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

FIGURE 3.8 – The F1 score and coverage for different similarity thresholds for *Rattus norvegicus* reference proteome

their InterPro domain signatures. We only used InterPro domain signatures for the purpose of EC annotation.

EC Annotation Performance Analysis

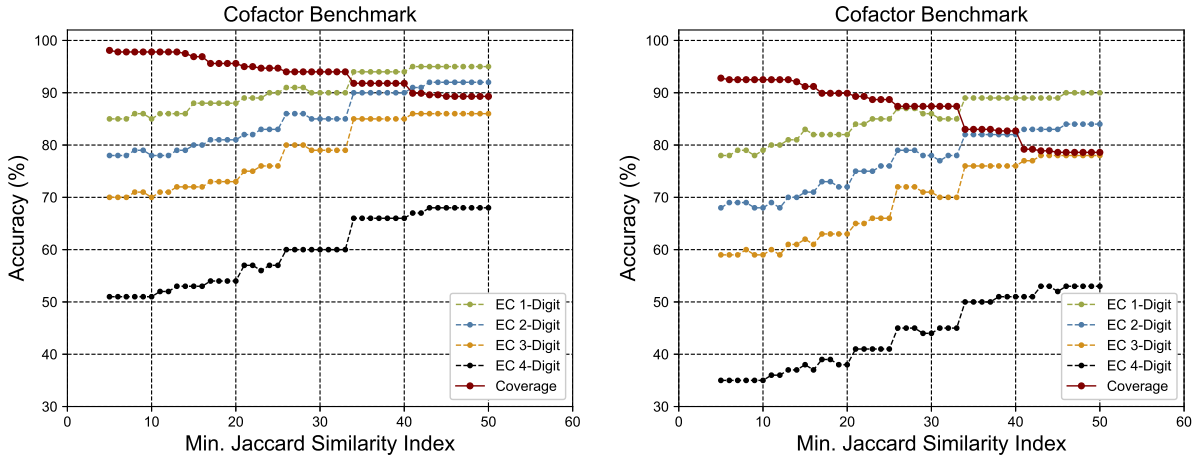
To validate the annotation performance of GrAPFI, we computed the accuracy, macro-precision, macro-recall, and macro-F1 score at different levels of EC number. For each query sequence, we picked the top-ranked annotation only. The validation method we have used is similar to a leave-one-out-cross-validation. For each proteome, when annotating a protein, we have removed that protein from the training set so that a direct mapping is not present. We also present a 10-fold cross validation for enzyme vs. non-enzyme classification (Fig-3.14 and 3.13).

The following formula (as used in [Li et al., 2018]) is used to compute the evaluation scores :

$$accuracy(y, y') = \frac{1}{N}\sum_{i=0}^{N-1} 1(y_i = y'_i),$$

Here, y and y' are the list of ground-truth and predicted annotations, respectively, and N is the size of the tested protein dataset. As EC numbers are hierarchical with 4 levels, we report level-wise precision, recall and F1-measure. Level-1 denotes main class, level-2 denotes sub-class, level-3 denotes sub-sub-class and level-4 denotes sub-sub-sub class. The accuracy is computed for each level of EC annotation. For evaluation purposes, we split the 4-digit EC annotation into its constituent parts. Then, for level-1 we consider first digit, for level-2 we take first 2 digits, for level-3 we take first 3-digits and finally for level-4 we take all four digits together. As the problem is a multi-class classification problem, we computed class-wise macro-precision, macro-recall, and macro-F1 score as follows :

$$Macro - precision(y, y') = \frac{1}{|M|}\sum_{l \in M} precision(y_l, y'_l),$$



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

FIGURE 3.9 – The accuracy and coverage for different similarity thresholds for *COFACTOR* benchmark proteins

$$Macro - recall(y, y') = \frac{1}{|M|} \sum_{l \in M} recall(y_l, y'_l),$$

$$Macro - F1(y, y') = \frac{1}{|M|} \sum_{l \in M} F1\ measure(y_l, y'_l),$$

Here, y_l is the part of y corresponding to label l and y'_l is the part of y' corresponding to label l. And M is the set of labels. In general the precision, recall, and F1-Measure are computed as follows when two sets A and P are given :

$$precision = \frac{|A \cap P|}{|P|},$$

$$recall = \frac{|A \cap P|}{|A|},$$

$$F1 - measure = \frac{2 \times precision \times recall}{precision + recall}.$$

Here, A and P are the sets of ground-truth and predicted annotations, respectively.

We also report coverage which is calculated according to $Coverage = K/T$, where T is the total number of proteins in the test set and K is the number of proteins for which at least one EC is predicted. For each query sequence, we consider the top-ranked annotation.

For the validation dataset, GrAPFI runs by setting different minimum Jaccard similarity index ranging from 0.05 to 0.5, and setting an upper limit of the similarity to 1 or less than 1.

Figures 3.6, 3.7, and 3.8 show the GrAPFI performance in terms of precision, recall, f1 measures respectively for the reference proteome of *Rattus norvegicus* for different similarity thresholds. Sub-figures 3.6b, 3.7b, 3.8b shows the precision, recall, F1 measures when the exact match in domain composition is not considered. Whereas, sub-figures 3.6a, 3.7a, 3.8a show the GrAPFI performance when the exact match in domain composition is taken into account. For other proteomes, the results are put in Appendix 1.

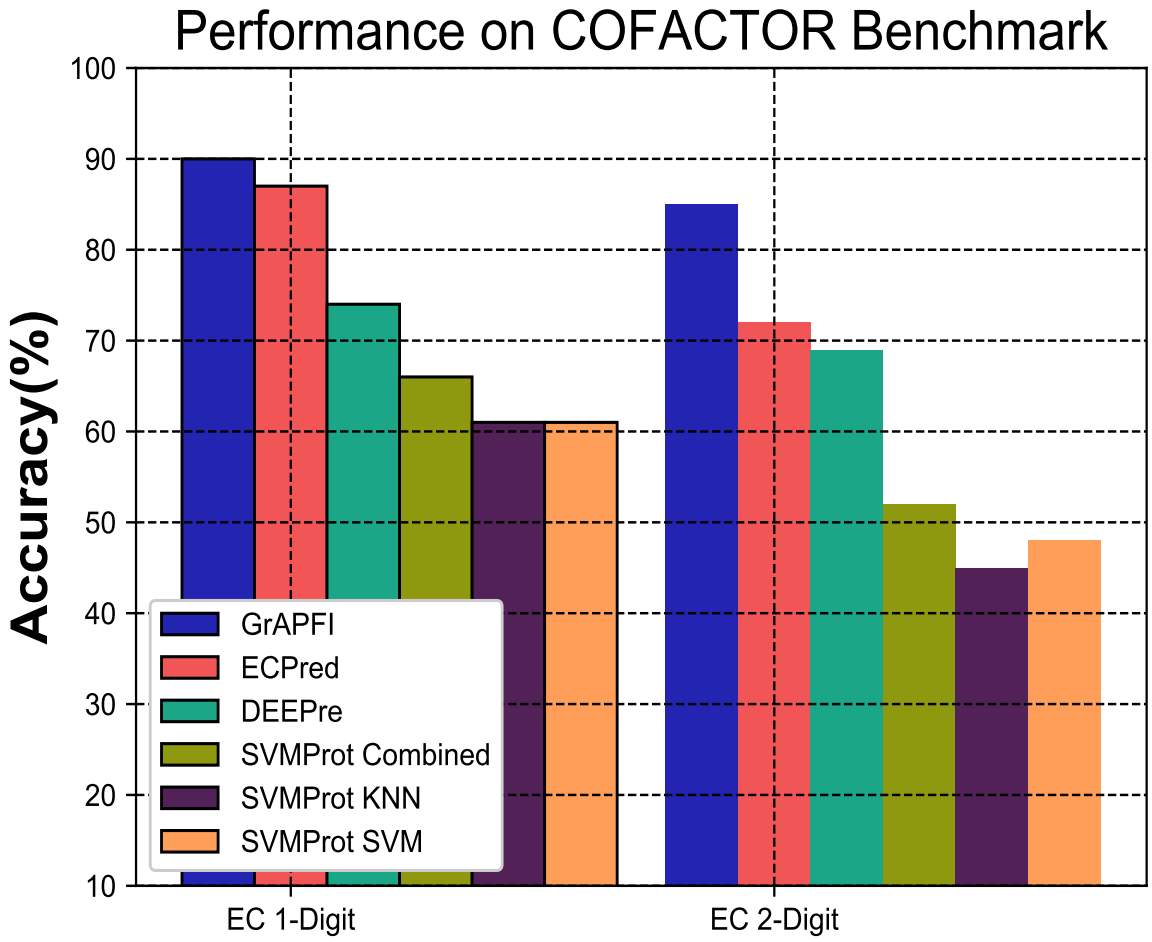


FIGURE 3.10 – Performance comparison of GrAPFI with SVMProt (SVM, KNN, and Combined), DEEPre, and ECPred for 2-digit EC number predictions.

In Figure 3.9a and 3.9b , we also show the performance of GrAPFI on COFACTOR benchmark dataset for various Jaccard domain similarity index ranging from 0.05 to 0.5, and setting an upper limit of the similarity to 1 and less than 1 respectively.

In these figures, we show the annotation accuracy (Y axis) against different Jaccard similarity thresholds (X axis) for the respective proteomes. We have considered similarity thresholds ranging from 0.05 to 0.5 as the annotation coverage falls significantly after 0.5. For each of the thresholds, we present the accuracy for EC-1, EC-2, EC-3 and EC-4 digit prediction shown in green, blue, orange and black color respectively. Along with accuracy, we also present the coverage of annotation (red curve). For each of the figures, we have two parts. The first part shows the accuracy and the coverage considering only the neighbors who have a Jaccard similarity of smaller or equal to 1. The second part considers the Jaccard similarity of less than 1. It can be seen from these figures that GrAPFI performs very well for all of the cases with a good coverage.

To compare GrAPFI with other state-of-the-art methods, we considered three machine learning based methods, namely ECPred [Dalkiran et al., 2018], DEEPre [Li et al., 2018], and SVMProt [Li et al., 2016]. The performance is compared based on the COFACTOR [Zhang et al., 2017] benchmark having 318 sequences.

The SVMProt prediction results cover three different conditions : (i) using SVM only, (ii)

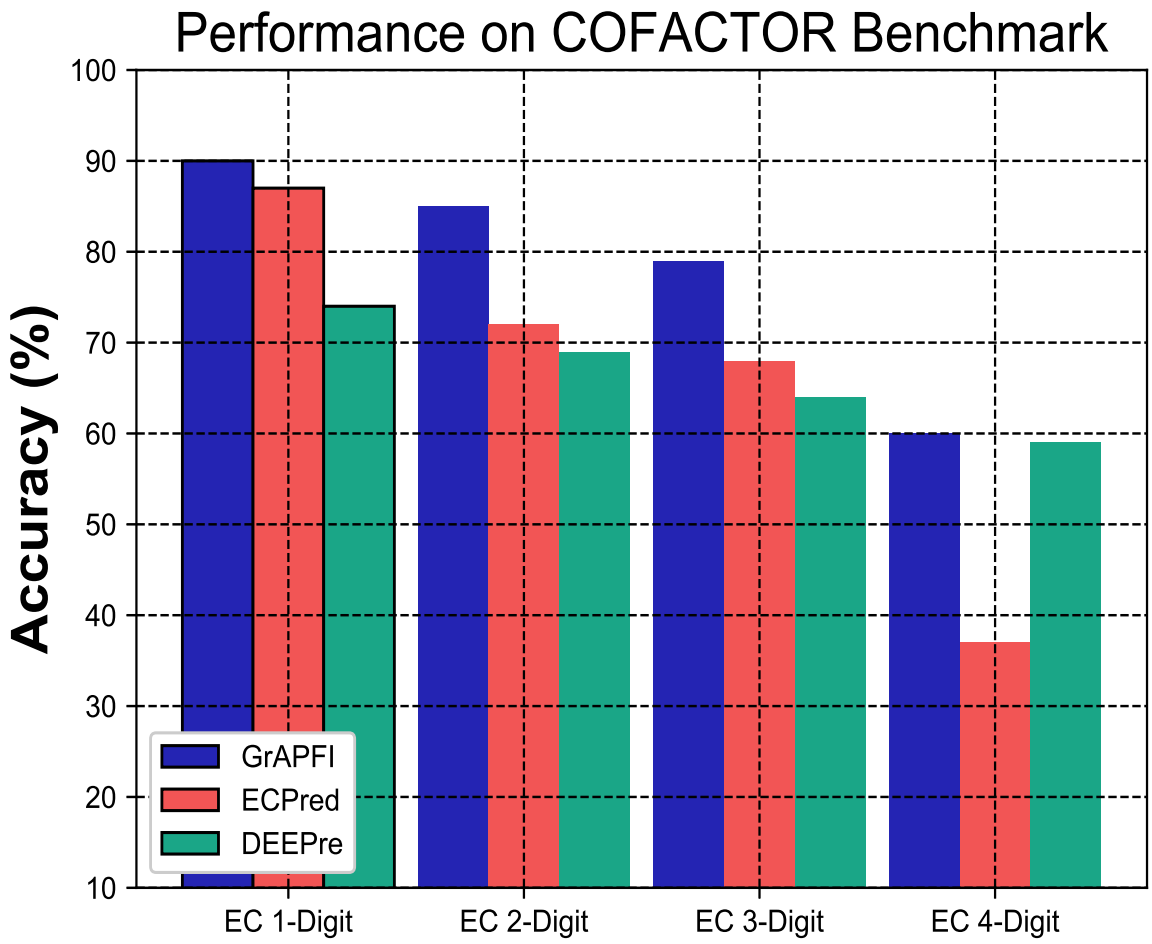


FIGURE 3.11 – Accuracy comparison of GrAPFI with DEEPre and ECPred for all 4 level of EC prediction.

using KNN only, and (iii) using SVM, KNN and PNN combined. Figure 3.10 shows the performance analysis for EC level-1 and EC level-2 prediction. The results presented here are achieved using a lower Jaccard similarity index of 0.3 and upper similarity index of 1.0. A much lower similarity threshold brings false positives that significantly reduce the accuracy. Based on the obtained results, a similarity threshold of 0.3 achieves a good trade-off between accuracy and coverage. Because not all of the methods can make predictions for all four EC levels, we compared GrAPFI only with ECPred and DEEPre for the 4 levels of EC numbers as shown in Figure 3.11. In Figure 3.12, we show the annotation coverage of the methods considered here. It shows that ECPred has greater coverage compared to other methods. However, GrAPFI is at the 2nd place (94% versus 98% for ECpred). The reason why GrAPFI fails to achieve the highest coverage is due to the fact that it is a neighborhood-based annotation method. GrAPFI performs label propagation by filtering out weakly linked neighbors determined by a minimum similarity threshold. Due to this filtering action, for a few cases, GrAPFI fails to suggest any appropriate annotation for query proteins. This reduces the total annotation coverage. However, on the other hand, GrAPFI increases the accuracy by considering strongly linked neighbors. As shown in Figures 3.11 and 3.12, GrAPFI has better accuracy compared to ECPred and DEEPre, but it gives slightly less coverage than ECPred.

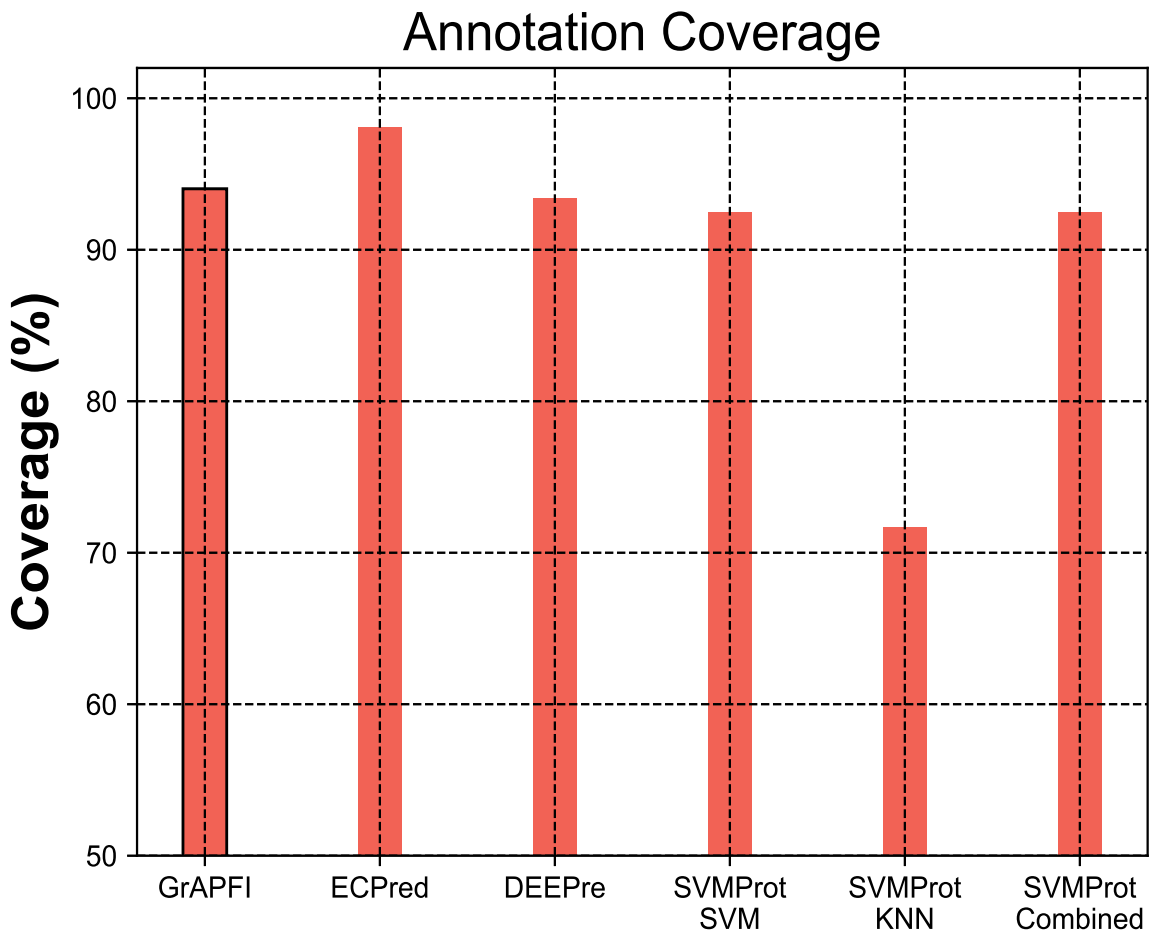


FIGURE 3.12 – Coverage of the considered methods.

3.1.5 Enzyme vs. non-enzyme classification

GrAPFI can be used in Enzyme vs. Non-enzyme classification task in a similar fashion as described in the above section. However, the training graph must include non-enzyme proteins. To experiment with enzyme vs. non-enzyme classification, we have used a well defined dataset of enzyme and non-enzyme proteins curated from UniProtKB [The UniProt Consortium, 2015]. This dataset is called "NEW" and was constructed as described in [Li et al., 2018] :

1. The SWISS-PROT (released on September 7, 2016) database was separated into enzymes and non-enzymes based on their annotation.
2. To guarantee uniqueness and correctness, enzyme sequences with more than one EC number or incomplete EC number annotation were excluded.
3. To avoid fragment data, enzyme sequences annotated as 'fragment' or with less than 50 amino acids were excluded. Enzyme sequences with more than 5000 amino acids were also excluded.
4. Redundancy bias is removed using CD-HIT [Fu et al., 2012] with 40% similarity threshold to sift the raw dataset, resulting in 22,168 low-homology enzyme sequences.
5. To construct the non-enzyme part, 22,168 non-enzyme protein sequences were randomly

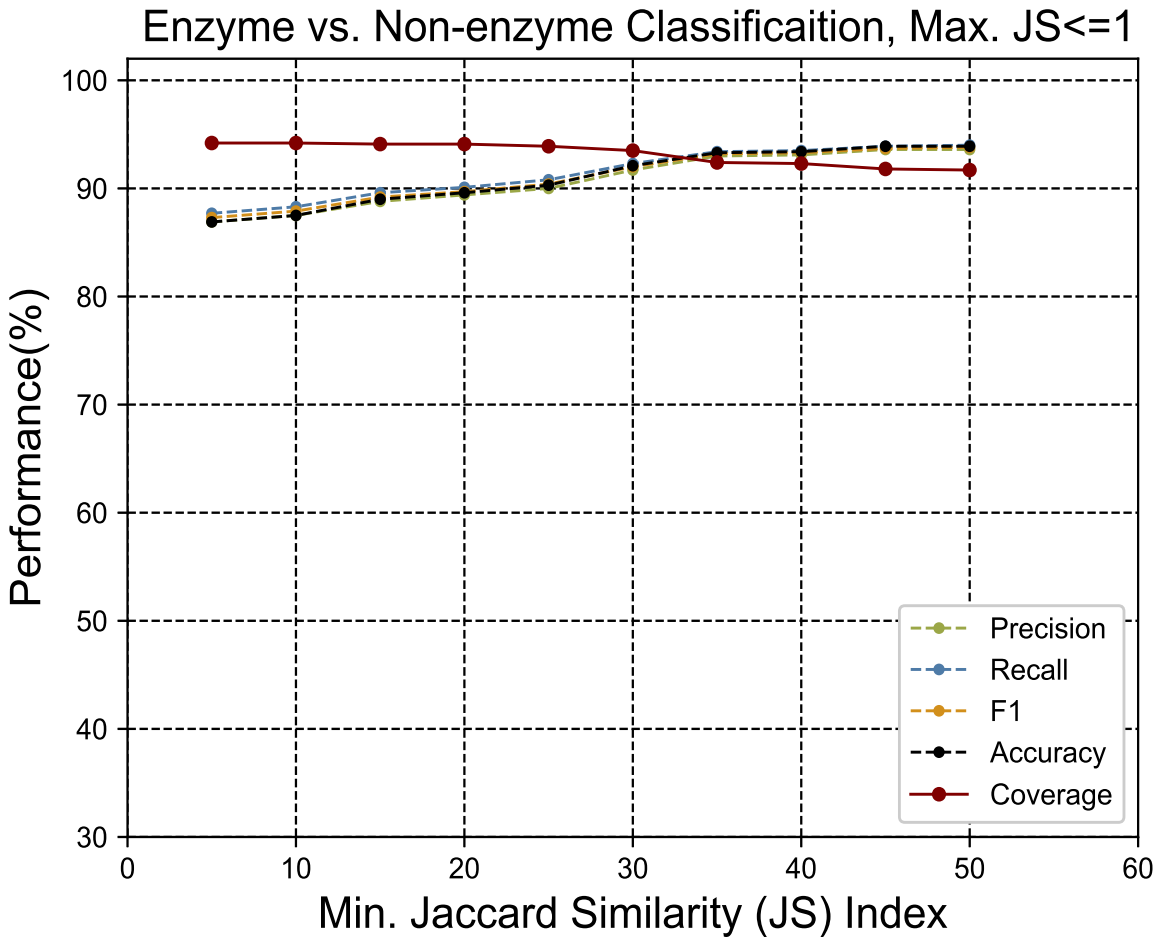


FIGURE 3.13 – The precision, recall, F1, accuracy and coverage score for various minimum Jaccard similarity index for the Enzyme vs. Non-enzyme classification using upper similarity index of 1.

collected from the SWISS-PROT (released on September 7, 2016) non-enzyme part, which were also subject to the above (2 to 4) steps. Thus the original dataset contains 22,168 enzymes and an equal number of non-enzymes.

The dataset contains the protein sequences along with their respective EC annotations. We have run InterProScan5 [Jones et al., 2014] to identify the domains contained in the sequences. Later, with the domain information, we have built the training graph. This graph contains 40040 proteins with 54% enzymes and 46% non-enzymes connected based on their domain composition.

To evaluate the annotation performance, we performed 10-fold cross validation on the training graph and average macro-precision, macro-recall, macro-F1 scores are computed for various Jaccard similarity indices. The result shows performance of enzyme vs. non-enzyme classification only. The experimental outcomes are shown in figures 3.14 and 3.13.

It is evident from the experimental outcome that GrAPFI can distinguish enzyme and non-enzyme proteins with a good score in all evaluation metrics. However, the coverage goes down as we move towards higher minimum similarity thresholds. One of the things to be noted that considering exact similarity match does not change the performance significantly as can seen in Figure 3.13.

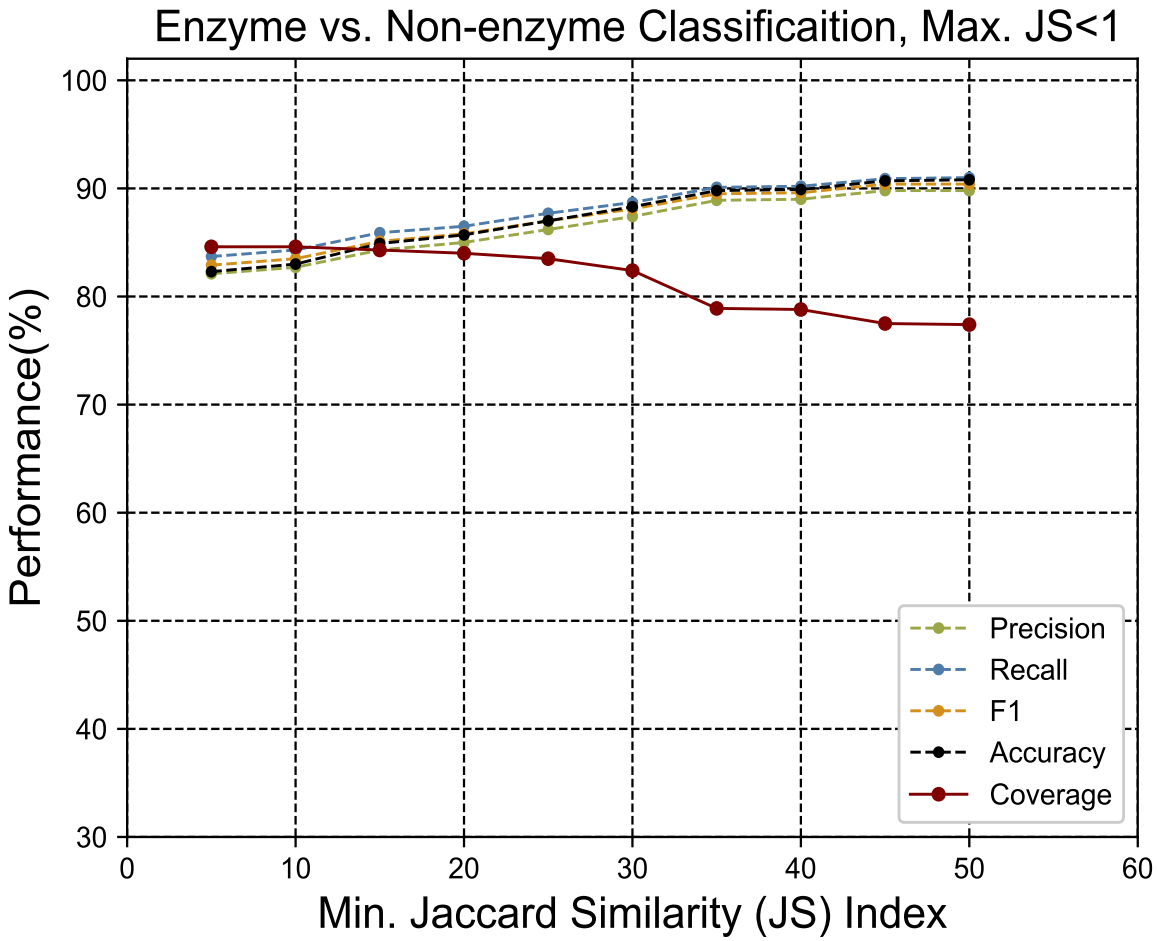


FIGURE 3.14 – The precision, recall, F1, accuracy and coverage score for various minimum Jaccard similarity index for the Enzyme vs. Non-enzyme classification using upper similarity index of less than 1.

3.2 Conclusion

Automatic protein function annotation is an important topic in the field of bioinformatics because of the lack of annotation of proteins due to high costs and time-consuming nature of manual functional annotation procedures. In this chapter, we introduced a neighborhood-based annotation technique for automatic functional annotation of proteins. We show the effectiveness of the method in EC number annotation based on its performance in annotating proteins from various well studied proteomes. In section 3.1, we explore new ways of connecting proteins. The proteins are connected based on domains that are potentially linked to the protein functions. This eventually means that GrAPFI is a biologically meaningful approach. One of the major advantages of using GrAPFI to annotate proteins is that it produces explainable high-quality annotations with a relatively simple annotation pipeline. The potential is evident from the experimental results. Although GrAPFI performs well, there are few drawbacks of using GrAPFI. For example, GrAPFI works on domain composition that can be achieved using another tool. GrAPFI can not be used with proteins lacking any domain information. And also for the proteins with single domain, in most cases, GrAPFI fails to find appropriate annotation. The reason

for this failure is that for a single domain protein, it is highly unlikely that there will be any strongly linked neighbors that can share annotations. This eventually left the protein without any labels or wrong one. In any case, if GrAPFI fails to find an annotation, it is possible to identify the reason behind the failure and it restricts itself from predicting any annotation. This behavior reduces the false positives. However, from the experiment, it is evident that GrAPFI performs with high annotation coverage. Unlike other hierarchical classification models like EC-Pred [Dalkiran et al., 2018] and DEEPre [Li et al., 2018], GrAPFI does not learn model for every class. Instead, it builds a giant network of proteins and applies label propagation for each query proteins. During annotation propagation, GrAPFI only considers the neighbors that are already annotated. The described approach could be easily distributed in order to handle large protein databases. The method is scalable for larger dataset using big data processing frameworks like Hadoop/Spark. We therefore aim to extent GrAPFI to use a distributed processing framework for the large scale annotation of the entire UniProtKB/TrEMBL database. Moreover, there is still scope of improvement specially for level-3 and level-4 predictions.

4

GrAPFI Improved by Semantic Similarity for Gene Ontology GO Annotation

In this chapter, we present a Gene Ontology annotation technique. We have designed a post-processing and pruning technique that works on GO Graph and computes semantic similarity among the GO terms to find the membership score. We have extended the GrAPFI approach, introduced in the previous chapter to be applied in the setting of GO annotation.

4.1 Gene Ontology (GO) Annotation and Semantic Similarity

Here, we apply GrAPFI, a graph-based Automatic Protein Function Inference approach for Gene Ontology(GO) annotation. We propose a pruning and hierarchical post-processing to eliminate the outlier annotations based on functional similarity discussed in GOGO [Zhao and Wang, 2018]. More specifically, our contributions are the followings :

— We extend GrAPFI to perform Gene Ontology Prediction. GrAPFI is a neighborhood-based label propagation approach that works on a network of proteins connected using domains and family information. GrAPFI was originally proposed for Enzymatic protein function prediction using Enzyme Commission (EC) Number.

— We integrate semantic similarity to take into account the hierarchical nature of the Gene Ontology data and to prune outlier annotations based on their distance in the semantic space. To find functional similarity, we used GOGO [Zhao and Wang, 2018] which is claimed to be a fast and efficient way of computing GO term similarity.

— We experimentally evaluate the performance of the proposed approach by annotating protein sequences with GO terms and report a comparative understanding of the efficacy of the proposed pruning technique for GO term prediction.

4.1.1 Function Annotation Using GrAPFI

GrAPFI (described in Chapter 3) is a neighborhood-based label propagation approach that works on a network of proteins connected using domains and family information. GrAPFI follows the following steps to perform function annotation :

1. First, we construct a graph using the protein information. Each node u of the graph represents a protein. An edge (u, v) between two nodes/proteins u and v means that the linked proteins share some attributes like domains and functional sites. A node u may have a set of labels $L(u)$ (one or more annotations to propagate), has a set of neighbors $N(u)$, and for every neighbor $v \in N(u)$, it has an associated weight $W_{u,v}$. Jaccard similarity is used to compute the link weight and computed as $W_{P1,P2} = \frac{|D1 \cap D2|}{|D1 \cup D2|}$ for two protein P1 and P2 having sets of domains $D1$ and $D2$, respectively.
2. Then, a label propagation approach is applied to the protein graph in order to infer functional properties of the unlabeled nodes. Given a query protein, based on the domains and family information of it contains, all the neighboring proteins and their annotations are retrieved from the weighted graph. After getting the neighbors, each label of the neighbors are weighted with the edge weights that these neighbors exhibit with the query protein. When retrieving neighbors, it is possible to select only those neighbors which meet a certain similarity threshold. This means that the links can be filtered based on a predefined cut-off weight. For each candidate annotation, GrAPFI provides a confidence score, namely model score (MS) that is computed as :

$$MS(u, i) = \frac{\sum_{v \in N(u)} W_{u,v} \sum_{j \in L(v)} \delta(j, i)}{\sum_{v \in N(u)} W_{u,v}} \tag{4.1}$$

where $MS(u, i)$ is the weighted score of the candidate function i for the query protein u. And $\delta(j, i)$ is 1 if the function j of the protein v is the same as the candidate function i, otherwise, 0.

4.2 Pruning Prediction Set Using Functional Similarity Score

We observed that the state-of-the-art tools in the field of GO annotation [Jiang and et al., 2016, Radivojac and et al., 2013] yield a large number of predictions. Due to the large number of predicted annotations for each protein, precision of the model declines while recall increases. However, results from these approaches raise a big concern on false positives in the predictions. Therefore, we need a method that increases the precision of the model, and hence decreases false positives in the predicted set.

To reduce the number of false positive annotations, we adopted a naive pruning technique by identifying and eliminating the outlier annotations using semantic similarity. Measuring semantic similarity between GO terms has always been an essential step in functional bioinformatics research. In a set of predicted GO annotations for a protein, pairwise semantic similarity between GO terms can show how closely these terms are related to each other and not just to the protein. We used an open-source tool called GOGO [Zhao and Wang, 2018] for calculating the functional similarity score between GO terms and thus used it to compute the membership score of each predicted GO terms.

GOGO is a relatively fast method which does not need to calculate the information content (IC) from a large gene annotation corpus and it considers the number of children nodes in the GO DAGs when calculating the semantic contribution of an ancestor node toward its descendent nodes. GOGO is based on GO DAG topology instead of IC which means that it is comparatively stable.

Given $DAG_g = (g, T_g, E_g)$ be the Directed Acyclic GO Graph of a term g and its ancestors

T_g, the weight of semantic contribution is calculated as,

$$w_e(t) = 1/(c + nc(t)) + d \tag{4.2}$$

Where, c and d are constants determined by empirical observations, $nc(t)$ is the total number of children of the term $t \in T_g$. And E_g is the set of edges of the links among the terms in T_g. The semantic contribution of each term in $DAG_g = (g, T_g, E_g)$ is defined as,

$$S_g(t) = \begin{cases} 1 & if \;\; t = g \\ max\{w_e(t) * S_g(t') | t' \in children(t)\} & if \;\; t \neq g \end{cases} \tag{4.3}$$

Therefore, the aggregated semantic value for the term g is computed as,

$$SV(g) = \sum_{t \in T_g} S_g(t) \tag{4.4}$$

In the case of two terms where $DAG_{g1} = (g1, T_{g1}, E_{g1})$ of term g1 and $DAG_{g2} = (g2, T_{g2}, E_{g2})$ of term g2, the semantic similarity between them is as follows :

$$SS(g1, g2) = \frac{\sum_{t \in T_{g1} \cap T_{g2}} (S_{g1}(t) + S_{g2}(t))}{SV(g1) + SV(g2)} \tag{4.5}$$

Finally, the functional similarity between a set of GO terms, A= {g1, g2, g3,..., gm} and a query GO term $g \notin A$ and is as follows :

$$SS(g, A) = max_{1 \leq i \leq m}(SS(g, g_i \in A)) \tag{4.6}$$

Once the semantic similarity between each pair of GO terms in the predicted set is calculated, we measure the membership of each annotation in the set. $SS(g, A)$ can be used to find the membership score of a particular GO term in a set of predicted GO annotations. Equation 4.5 is reused to compute the membership score as follows :

$$SS(g_i, A) = max_{1 \leq j \leq m}(SS(g_i \in A, g_j \in A \setminus \{g_i\})) \tag{4.7}$$

Where, $SS(g_i, A)$ denotes the membership score of term g_i in a set of terms A.

Instead of maximum, membership score can also be calculated as the average and Root Mean Square (RMS) score of each annotation in the set. For this study, we used RMS score as it gave the best results. We name this measure of membership as semantic similarity (SS) score.

4.3 Aggregation of Scores

In all state-of-the-art GO annotation models, used for experiments in this study, there is a prediction score associated with each predicted annotation for every protein. We refer to this as model score (MS).

For a protein, u with a set of predicted annotations A, each annotation $g \in A$ has two scores associated to it : 1) first, the Model Score (MS) , defined as $MS(u, g)$, which shows the credibility with which the annotation was predicted by a particular annotation tool and 2) second, the Semantic Similarity (SS) score, defined as $SS_u(g, A)$, which shows the semantic similarity of each member annotation g to the predicted set A. Now, we need to combine these scores to find a combined prediction (CP) score, defined as $CP_u(g, A)$, for each annotation $g \in A$ of

protein u. Joining the scores into a single score provides an overall assessment. A score should be able to distinguish between annotations that score average in both MS score and SS score, from those that score high in one scoring scheme and low in the other scheme. Therefore, instead of averaging the scores, we follow the following scheme :

$$CP_u(g, A) = \sqrt{\frac{(\frac{MS(u,g)}{max\_MS})^2 + (\frac{SS_u(g,A)}{max\_SS})^2}{2}} \tag{4.8}$$

Here, max_MS and max_SS denotes the maximal model score and semantic similarity score, respectively. Range for both the scores is from 0 to 1 and hence are bounded. Since this is a technique to prune an already predicted set, we take square root in the equation to increase the overall value of combined scores so as to increase threshold cutoff. Once we have the combined score, we can take a certain score as cutoff to filter the predicted set. Annotations with scores above the cutoff forms a new predicted set.

The final step of the process is hierarchical post-processing of predictions in the new predicted set. In the Gene Ontology DAGs, the GO terms holds different parent-child relations putting biologically closer GO terms hierarchically nearer in the graph. We implemented a methodology to include more reliable predictions by including the ancestors of target GO term in the new set of prediction. The ancestors of a GO term in the DAG that the term belongs to, have a very high semantic similarity with the term. Therefore, we first topologically sorted the DAG for each GO category and determined all possible paths from each GO term to the root of the corresponding category. Finally, we follow these paths from terms to the root, one by one and add corresponding ancestors to the set of predictions to obtain final prediction set.

4.4 Experiments and Result Analysis

4.4.1 Datasets

To experimentally validate the performance of the proposed technique, we have used a benchmark test set published in MetaGo [Zhang et al., 2018]. For GrAPFI, we build the network using the training data from CAFA3 [5]. CAFA3 is a well known competition that seeks to annotate a list of protein sequences waiting for proper annotation. Along with target sequences, CAFA3 also published sequences that can be used as training data to develop models. In this study, to build the network, we have used CAFA3 training sequences and we have collected domain and family information for those proteins from UniprotKB. After that, we have built the graph of CAFA3 training proteins. This graph contains more than 65,000 nodes as proteins and an average 16 ground-truth GO terms per protein. To prepare the test set, we have used MetaGO benchmark sequences and run InterProScan [Jones et al., 2014] to identify the domains and family information from sequence. Using the domains and family information of test proteins, we run GrAPFI and other annotation tools over all of the test proteins and annotated them with appropriate GO terms post-processed using proposed approach.

4.4.2 Result Analysis

In the Table 4.1, we show the annotation performance using standard evaluation measure namely precision, recall and F1 score. Among the top performing methods, only a few had their source code available to run experiments. Therefore, we focus on three easily available tools

5. https ://www.biofunctionprediction.org/cafa/

namely GrAPFI, PANNZER and DeepGOPlus [Kulmanov and Hoehndorf, 2020], an improved version of DeepGO. DeepGoPlus learns models with less parameters than DeepGO. These tools are recently published and claimed to be high performing. We show that the semantic similarity improves the precision by many folds. However, the approach suffers from low recall as the number of predictions is much lower than the original predictions. This reduced number of predictions per protein essentially reduces the recall score resulting in lower F1-max scores.

TABLE 4.1 – The experimental results for cases when 1) No-post-processing : without post-processing and pruning 2) SS-max : post-processed using highest semantic similarity score as cut-off, 3) SS-5 : post-processed using 5th highest SS score as cut-off and 4) SS-5-MS-max/2 : post-processed using 5th highest semantic similarity and (maximum model score)/2 as cut-off

Method	Post-processing cut-off	Precision	Recall	F1-max
GrAPFI	No-post-processing	0.165	0.108	0.107
	SS-max	**0.573**	0.115	0.175
	SS-5	0.445	0.380	0.376
	SS-5-MS-max/2	0.440	**0.391**	**0.379**
Pannzer	No-post-processing	0.547	**0.942**	**0.668**
	SS-max	**0.637**	0.225	0.301
	SS-5	0.634	0.515	0.536
	SS-5-MS-max/2	0.603	0.689	0.609
DeepGOPlus	No-post-processing	0.053	**0.653**	0.095
	SS-max	**0.249**	0.120	0.138
	SS-5	0.186	0.182	0.160
	SS-5-MS-max/2	0.167	0.233	**0.1725**

We run the above mentioned annotation tools on MetaGo benchmark data and obtain results of annotation prediction. These predicted sets are further pruned using semantic similarity and hierarchical post-processing and results are mentioned in Table 4.1. Semantic similarity score and hierarchical post-processing score are obtained for each prediction for each protein. Different cut-offs of these two scores along with the score obtained from the model is used for analysis. For each annotation tool, Table 4.1 shows the annotation outcome in four cases : 1) without any kind of pruning and post processing, 2) when highest semantic similarity score is the cutoff, 3) when 5th highest semantic similarity score is the cutoff and 4) when 5th highest semantic similarity score and half of the maximum model score are the cutoff. From the Table 4.1, it is evident that the proposed post-processing and pruning techniques that use the semantic similarity of predicted GO terms improves the overal performance in most cases. In particular, it improves the precision by many folds. For example, the precision of GrAPFI is improved from 16.5% to 57.3% using maximum semantic similarity score as cut-off during post-processing. Similarly, the proposed combined scoring improves the precision of Pannzer and DeepGOPlus by many folds.

4.5 Conclusion

Here, we extended the GrAPFI described in Chapter 3 to use it in Gene Ontology term annotation. We propose a hierarchical post-processing and pruning techniques based on the semantic similarity of GO terms in GO ontology. There are a number of tools that exist to perform automatic protein function annotation using GO terms, EC numbers, ligand binding sites etc.

These tools use various attributes and different methods to accomplish the task. Although they show higher performance based on F1 score, it is clear that this high F1-score is coming from a higher recall as they predict a large number of candidate annotations. This, in turn, increases the number of false positive annotations. In this section, 1) we present a graph-based protein function inference method extended for GO term prediction, and 2) we propose an efficient pruning and hierarchical post-processing technique by integrating semantic similarity of candidate annotations. We experimentally validate that the proposed method can significantly improve the annotation outcome. In fact, in most cases, recall is significantly low as the number of annotations is fewer compared to the number of annotations predicted by other tools. Nevertheless, the precision is improved by many folds as we select the highly coherent semantically close annotations.

Troisième partie

Representation Learning for Biological Entities

5

Functional Annotation of Protein Using Domain Embedding Based Sequence Classification

In this chapter, we present an automatic functional annotation technique using neural-network-based word embedding exploiting domain and family information of proteins. Domains are the most conserved regions in protein sequences and constitute the building blocks of 3D protein structures. To do the experiment, we used fastText [6], a library for learning of word embeddings and text classification developed by Facebook's AI Research lab. The experimental results show that domain embeddings perform much better than k-mer embeddings.

5.1 Text Classification

One of the important tasks in natural language processing is to classify text into classes such as tags, categories, labels, and so on. Text classification is widely used in web search, information retrieval, ranking and document classification. Due to recent successes, neural network based models are prevalent in text classifications. Although the representation capability of neural networks is higher, training neural network based deep learning models is computationally expensive due to the presence of non-linear hidden layer [Mikolov et al., 2013, Joulin et al., 2017]. On the other hand, linear classifiers are simple and efficient, yet achieve better performance in the context of text classification. However, linear classifier like SVM does not share parameters among features and classes [Joulin et al., 2017], which eventually limits the generalization power of linear classifier. The fastText [Joulin et al., 2016] text classification tool uses log-linear model with a shallow neural network to build a simple, fast, and efficient text classifier with word embedding.

5.2 Text Classification for Protein Function Annotation

Natural language text possesses a defined linguistic structure containing an array of words delimited by various punctuation marks. By analogy, biological texts such as protein sequences

6. https ://github.com/facebookresearch/fasttext

are strings of letters selected from an alphabet consists of 20 letters, each representing an amino acid [Kimothi et al., 2016]. Essentially one string stands for a single protein. Unlike natural texts, there is no way of formally defining words or phrases in protein sequences. Therefore, using a text classification model requires further pre-processing of protein sequences. The most common way of pre-processing is to break the sequences into biological words commonly known as K-mers that are smaller units of size k composed of consecutive alphabets. The pre-processing can be done in two different ways : 1) Overlapping k-mers achieved by moving a k-size window over the sequence. For example, let us break "$MAPPSVFSEV$" into overlapping 3-mers. The corresponding 3-mers are MAP, APP, PPS, PSV, SVF, VFS, FSE, and SEV. Therefore, the biological sequence "$MAPPSVFSEV$" is transformed into following space delimited text : *MAP APP PPS PSV SVF VFS, FSE, SEV* ; (2) Using non-overlapping k-mers, k numbers of sequences are generated by splitting the original sequences into non-overlapping words of k consecutive letters with a starting position moved by one letter for each newly generated sequence [Asgari and Mofrad, 2015, Kimothi et al., 2016]. For example, for the sequence "$MAPPSVFSEV$", considering 3-mers, the 3 newly generated space delimited sequences are as follows :

1. *MAP PSV FSE*
2. *APP SVF SEV*
3. *PPS VFS*

Non-overlapping K-mers have been used for learning word embedding tasks and have been shown to have better prediction accuracy when applied to family classification task [Asgari and Mofrad, 2015]. Overlapping K-mers are widely used in homology based sequence search in large databases of protein sequences like [Altschul et al., 1997].

This paper presents a novel way to tokenize the protein sequences for the purpose of functional annotation. Instead of k-mers, we use domain and family information of protein in order to learn protein domain embeddings. The rationale behind using domain information is that the domains are the meaningful units of protein sequence conserved across similar sequences. Domains may be considered as natural building blocks of proteins. Due to evolution, protein domains may have gone through changes such as duplication, fusion, recombination to produce proteins with distinct structures and functions [Kummerfeld and Teichmann, 2009]. On the other hand, k-mer words do not carry any biologically significant meaning by themselves. Thus, tokenizing a protein sequence into sentence of domains is more informative than sentence of k-mer words. The experimental results presented in experimental section verify the higher performance of domain embeddings.

Two steps are necessary to prepare a training corpus using domain information :

(1) Firstly, for each of the sequences, identify domain signatures using InterProScan [Jones and et al., 2014, Quevillon and et al., 2005, Mitchell et al., 2018a], which is a sequence analysis software that integrates different protein signature recognition methods into one resource. InterProScan provides domain signatures along with their location of appearance in the sequence.

(2) Secondly, InterProScan output is processed to collect the domain signatures and sorting them according to their location of appearance. The domains are organized in ascending order of their starting position in the sequence to form the domain-sentence. Thus each line of the final corpus is a list of domains found by InterProScan for a given protein sequence along with true EC labels.

In this paper, we propose an automatic protein function annotation technique that uses a shallow neural network based text classification method based on domain embeddings. To accomplish the task, We have used *fastText* developed by Facebook Artificial Intelligence Research team to train a supervised sequence classification model as well as domain embeddings. We show

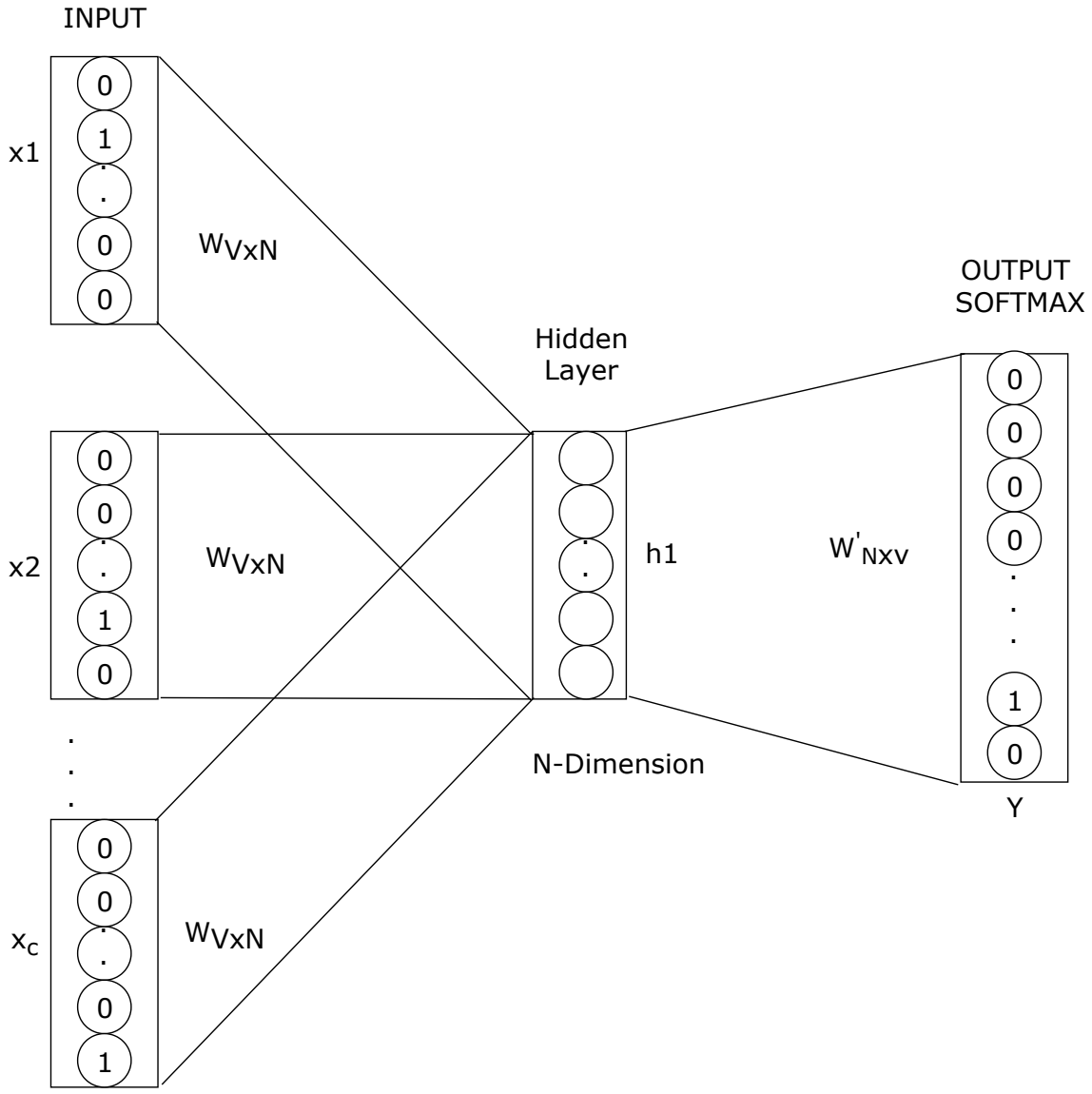


FIGURE 5.1 – Continuous Bag of Words Model Architecture [Mikolov et al., 2013].

a comparative analysis of overlapping 3-mers, non-overlapping k-mers and domain embeddings for protein function annotation with enzyme commission number.

5.3 Methods

Our proposed enzyme classification technique is based on supervised text classification model for natural language processing. The supervised classification model is designed in a similar way to the Continuous Bag of Words (CBOW) described in [Mikolov et al., 2013] architecture, where the target word is replaced by the EC annotation. The typical CBOW model architecture is shown in Fig. 5.1. CBOW employs very simple neural network with single hidden layer for learning the projection of individual word. In supervised mode, for N textual documents or sentences, the

following negative log-likelihood is optimized over the classes as described in [Joulin et al., 2017] :

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(f(W'Wx_n)),$$

Where W is a weight matrix that serves as a look-up table over the words connecting input to the hidden layer, W' is a weight matrix that connects hidden layer with output layer where softmax function f is applied to compute the probability distribution of the labels. x_n is the normalized bag of words of the n-th document. y_n is the label under consideration.

The discretely learned word representations are averaged to learn the text representation which is then fed into a linear classifier. The model uses stochastic gradient descent based back propagation for optimizing the loss function.

To apply this model to enzyme classification, we have used domain and family signatures of proteins as discrete words. The individual domain signatures found by running InterProScan against each protein sequence serve the purpose of words. While preparing the training corpus, the domains are arranged in ascending order of their location of appearance in the sequence. For the comparison purposes, we have also used overlapping and non-overlapping k-mer based biological words processed from sequence data. The Enzyme Commission (EC) numbers are used as labels without any further processing.

After pre-processing the protein sequences to generate domains and K-mers and associating appropriate EC labels, we prepared a large corpus of biological texts for the purpose of learning embeddings and classification models. The work-flow for the classification task is shown in Figure 5.2. This figure shows the steps involved in training a domain embedding model using fastText supervised learning. The raw sequence data is transformed into domain data using InterProScan [Jones and et al., 2014] sequence analysis tool, and then fed into a supervised learning model to learn the embeddings and classification model. Finally, the test data are fed into the model to predict the EC annotations.

5.4 Experiments and Result Analysis

In this Section, we first present the used data. Then, we present our experimental protocol and we discuss the obtained results.

5.4.1 Dataset and Training

To evaluate the method, we have used a well defined dataset of enzyme and non-enzyme proteins curated from UniProtKB [The UniProt Consortium, 2015]. This dataset is called “NEW” and was published by [Li et al., 2018] and was constructed by following the following rules.

1. The SWISS-PROT (released on September 7, 2016) database was separated into enzymes and non-enzymes based on their annotation.
2. To guarantee uniqueness and correctness, enzyme sequences with more than one EC number or incomplete EC number annotation were excluded.
3. To avoid fragment data, enzyme sequences annotated as ‘fragment’ or with less than 50 amino acids were excluded. Enzyme sequences with more than 5000 amino acids were also excluded.
4. Redundancy bias is removed using CD-HIT [Fu et al., 2012] with 40% similarity threshold to sift the raw dataset, resulting in 22,168 low-homology enzyme sequences.

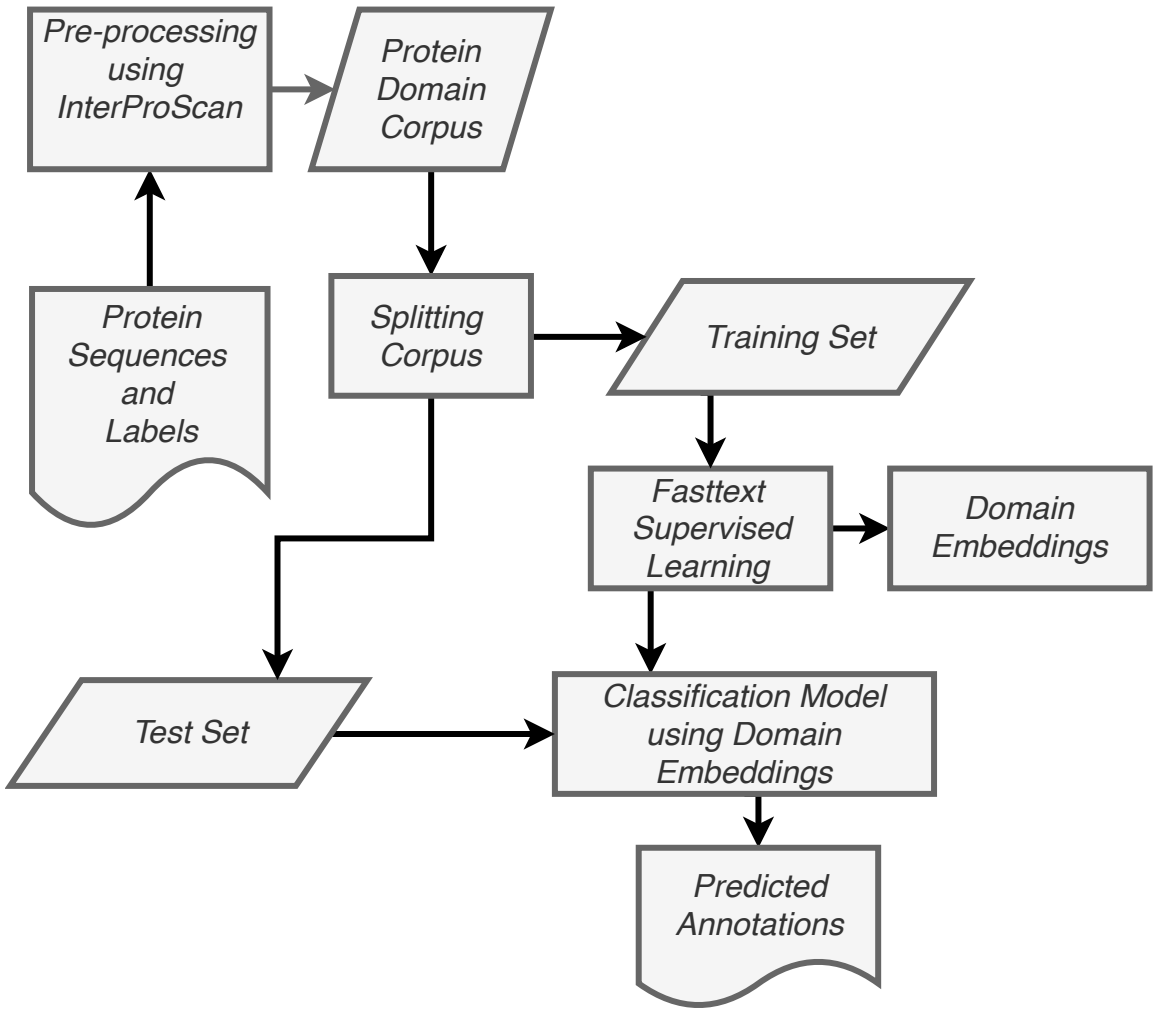


FIGURE 5.2 – Data preparation and training work-flow for Domain Embedding based Protein Function Annotation.

5. To construct the non-enzyme part, 22,168 non-enzyme protein sequences were randomly collected from the SWISS-PROT (released on September 7, 2016) non-enzyme part, which were also subject to the above (2 to 4) steps. Thus the original dataset contains 22,168 enzymes and an equal number of non-enzymes.

To build the corpus for learning the embeddings, each sequence has undergone three different pre-processing treatment for 3 different types of sequence tokenization described in the section 5.2. For overlapping k-mer, each sequence is split into overlapping K-mer words. We have chosen 3-mer for our experiment meaning a window of size 3 is moved over the sequence to split it into overlapping 3-mers.

For non-overlapping k-mer, each sequence was transformed into 3 sequences of non-overlapping 3-mers. And same label is associated with all of the 3 sequences as they are generated from the same sequence with single EC label.

In the last case, we used domain and family signatures of proteins as words to build the training corpus. Each sequence was transformed into a list of domains using InterProScan

[Jones and et al., 2014] software of Version 5.35-74.0. We built 3 large corpus using the 3-mers (overlapping and non-overlapping) and domains and each sentence is associated with EC labels to train the supervised classification model. For non-enzyme, we have labelled them with "__label__NANZ".

For the implementation purpose, we used fastText [Joulin et al., 2016] which is a library for learning of embeddings, and text classification developed by Facebook's AI research.

5.4.2 Evaluation

To evaluate our method, we used 10-fold cross-validation on NEW Dataset. We split the text corpus into 10 parts, and considered one of them as a testing set with the remaining 9 parts composedly being used as the training set. The cross validation results show promising performance of the domain embedding-based Enzyme classification. For each query sequence, we picked the top ranked annotation only. To validate the performance, we computed the accuracy, macro-precision, macro-recall, and macro-F1-measure at different levels of EC number. The following formulae (as used in [Li et al., 2018]) were used to compute the evaluation metrics :

$$accuracy(y, y') = \frac{1}{N}\sum_{i=0}^{N-1} 1(y_i = y'_i),$$

Here, y and y' are the list of ground truths and predicted annotations. The accuracy is computed for each level of EC annotation. As the problem is a multi-class classification problem, we computed macro-precision, macro-recall, and macro-F1 score as follows :

$$Macro - precision(y, y') = \frac{1}{|M|}\sum_{l \in M} precision(y_l, y'_l),$$

$$Macro - recall(y, y') = \frac{1}{|M|}\sum_{l \in M} recall(y_l, y'_l),$$

$$Macro - F1(y, y') = \frac{1}{|M|}\sum_{l \in M} F1\ measure(y_l, y'_l),$$

Here, y_l is the part of y with the label l and y'_l is the part of y' with label l. And M is the set of classes. In general the precision, recall, and F1-Measure are computed as follows when two sets A and P are given :

$$precision = \frac{|A \cap P|}{|P|},$$

$$recall = \frac{|A \cap P|}{|A|},$$

$$F1 - measure = \frac{2 \times precision \times recall}{precision + recall}.$$

Here, A is the set of ground truths and P is the set of predictions. As EC numbers are hierarchical with 4 levels, we report level-wise precision, recall and F1-measure. Level-1 denotes main class, level-2 denotes sub-class, level-3 denotes sub-sub-class and level-4 denotes sub-sub-sub class. We add level-0 to the step of identifying enzyme or non-enzyme. For each query sequence, we pick the top-most annotation. For evaluation purposes, we split the 4-digit EC annotation into its constituent parts. Then, for level-1 we consider first digit, for level-2 we take first 2 digits, for level-3 we take first 3-digits and finally for level-4 we take all four digits together.

In our experiments, we have trained the model using 0.9 as learning rate, 150 hidden units, window size of 5 and trained for 200 epochs. We have used Hierarchical Softmax as loss function and word n-grams of 2. The cross validation result is shown in the table 5.1.

Table 5.1 – The experimental results of 10-fold Cross-Validation on "NEW" dataset

Level	Embeddings	Accuracy	Macro-precision	Macro-recall	Macro-F1
Level-0	3-mer Non-overlap	0.741	0.768	0.776	0.772
	3-mer overlap	0.738	0.816	0.825	0.82
	Domain	**0.943**	**0.968**	**0.97**	**0.969**
Level-1	3-mer Non-overlap	0.523	0.243	0.24	0.241
	3-mer overlap	0.529	0.251	0.245	0.248
	Domain	**0.929**	**0.918**	**0.913**	**0.916**
Level-2	3-mer Non-overlap	0.467	0.09	0.089	0.089
	3-mer overlap	0.47	0.101	0.097	0.099
	Domain	**0.918**	**0.823**	**0.812**	**0.817**
Level-3	3-mer Non-overlap	0.452	0.062	0.06	0.061
	3-mer overlap	0.455	0.073	0.07	0.071
	Domain	**0.909**	**0.757**	**0.757**	**0.757**
Level-4	3-mer Non-overlap	0.434	0.037	0.036	0.037
	3-mer overlap	0.439	0.052	0.05	0.051
	Domain	**0.851**	**0.62**	**0.633**	**0.626**

Table-5.1 shows the accuracy, precision, recall and F1 measure of the cross-validation performed over the NEW dataset. As EC numbers consist of four digits, we present the evaluation metrics for four levels (level-1, 2, 3, 4). To measure the efficiency in classifying enzymes and non-enzymes among the test sequences, we present level-0 accuracy, precision, recall and F1 measure. The result shows that the proposed domain embedding based classification can differentiate enzymes and non-enzymes with an accuracy of 94.3% which is better than k-mer based embeddings(73%-74%). Along with accuracy, we also report the macro precision, recall, and F1 score weighted over classes. Macro-{precision, recall and F-1} scores give a reliable measure for unbalanced data. Domain embeddings perform with very high precision and recall of 97% for level-0 prediction task. It is interesting to see that the macro-F1 score confirms the higher accuracy shown by the proposed domain embeddings.

For level-1 predictions, we predict the main class if the protein is an enzyme. There are 6 different main classes. We measure the level-1 accuracy as how accurately it can identify the non-enzymes and enzymes with the correct main class. Similar to level-0, we also present the class based macro-{precision, recall, and F1} scores as the test data is class imbalanced. The performance measures show that domain embedding based classification performs better in all metrics.

In a similar fashion, we also report the accuracy, macro-{precision, recall and F1} scores for level-2, level-3 and level-4 predictions. For all of the levels, our proposed annotation technique outperforms K-mer based embeddings. However, as we go to higher EC levels, the accuracy falls off because a higher EC level is very specific. For example, a Level-4 EC number describes an enzyme that is specific for a particular type of substrate molecule. However, the domain embeddings based classification shows promising performance for level-4 prediction also.

In summary, from the results shown in Table 5.1, it is evident that domain-based embeddings perform noticeably better than k-mer based word embeddings in all evaluation metrics and also for all levels of EC hierarchy.

5.5 Conclusion

Here we propose a novel protein function annotation approach using domain embedding-based sequence classification instead of k-mer based word embedding. To show the superior performance of the proposed method, we used 10-fold cross-validation on benchmark dataset. We measured the annotation performance using accuracy and we also report the macro precision, macro-recall, and macro-F1 measure to reduce the effect of class imbalance in the test dataset. According to all the evaluation metrics we considered, the proposed approach show better performance. One of the strengths of the proposed method lies in its simplicity. The method learns domain embeddings using a single-layer neural network. Due to the use of shallow neural network, the training is faster than other multi-layer deep networks. We have used hierarchical softmax loss function to make training even faster. Unlike other hierarchical classification models, for example, ECPred [Dalkiran et al., 2018] and DEEPre [Li et al., 2018], the proposed method learns single model instead of learning many models each for every class. The method is scalable for larger dataset using CUDA-based GPU units. Although the proposed method performs well, there is still scope of improvement, particularly, for level-3 and level-4 predictions. As a future goal, we envision to improve the method for more precise predictions, and also to apply the similar approach for protein function annotation using GO Terms.

6

Prot-A-GAN : Protein Annotation using GAN-inspired Knowledge Graph Embedding

Automatic protein function annotation is a challenging task in bioinformatics research. Without proper annotation, the use of the protein data can be very limited. Manual annotation by experts are expensive, slow, and insufficient to fill the gap between the annotated and unannotated proteins. Although sequences are the primarily available protein data, we leverage protein-profile information created by expert human curators. To connect distinct protein profile, one of the natural ways is to build information network of proteins. We adapted knowledge graph as it provides a robust way to connect heterogeneous information sources with many different types of relations among them. In this chapter, we present Prot-A-GAN (Protein Annotation GAN) : a generative knowledge graph embedding technique using GAN-like adversarial training for the purpose of protein function annotation. Following the terminologies of GAN : 1) we train a Discriminator with domain-adaptive negative sampling, and 2) we train a Generator as a random walk over the knowledge graph that identify paths between protein and GO annotations. The task of protein function annotation is performed by discovering links between query protein and candidate GO terms. We evaluate the method by performing protein function annotation using GO terms on human disease proteins from UniProtKB-SwissProt. As a proof-of-concept, the experiment shows promising outcome and opens up new avenue for further exploration, exclusively for protein function annotation. Although the Prot-A-GAN is designed for protein function annotation, it is equally applicable for other reasoning task involving knowledge graph and link prediction.

To be precise, followings are the contributions discussed in this chapter :

— We build a knowledge graph integrating different information related to proteins that are helpful to perform automatic protein function annotation.
— We formulate the problem of protein function annotation as a link prediction task over knowledge graph.
— We design, implement and evaluate Prot-A-GAN pipeline for automatic protein function annotation using biomedical knowledge graph.

6.1 Prot-A-GAN Framework

A knowledge graph can be defined as $G = \{\mathcal{V}, \mathcal{E}, \mathcal{R}, \mathcal{A}\}$ as a directed heterogeneous graph where,

1. $\mathcal{V}$ is a set of objects/entities of different types.
2. $\mathcal{R}$ is the set of relation types that connect the objects in $\mathcal{V}$. The links in the knowledge graph are mapped to corresponding relation types with type mapping function $\psi : \mathcal{E} \longrightarrow \mathcal{R}$.
3. $\mathcal{E}$ is the set of edges represented as triples of the form (s, p, o), where s : subject/head/source node, p : predicate/relation/type, and o : object/tail/destination node. Each link $e \in \mathcal{E}$ belongs to a link type (relation) $\psi(e) \in \mathcal{R}$.
4. The entities of the knowledge graph are mapped to their corresponding node types with a mapping function $\phi : V \longrightarrow \mathcal{A}$. $\mathcal{A}$ denotes the set of node types. Each entity $u \in \mathcal{V}$ belongs to an entity type $\phi(u) \in \mathcal{A}$.

$\mathcal{N}_s^p$ is the set of neighbors for node s for a particular relation p. Let us also consider that $f(s, p, o)$ with $s, o \in \mathcal{V}$, $p \in \mathcal{R}$ is a scoring function that estimates the likelihood of a triple to be a positive fact i.e. how likely the head s and tail o will form an edge in the knowledge graph. In this section, we will describe the training strategy for the proposed framework for knowledge graph embedding. Similar to GraphGAN [Wang et al., 2019], we have trained a GAN-like model to learn entity and relation representation instead of only node representation. Θ_G^V and Θ_G^R denotes d-dimensional entity and relation embeddings for Generator. Similarly, Θ_D^V and Θ_D^R denotes d-dimensional entity and relation embeddings for Discriminator.

Generative Adversarial Network (GAN) [Goodfellow et al., 2014] is a popular deep learning technique in computer vision to produce realistic images, faces and styles. A typical GAN has two main components : 1) Generator, and 2) Discriminator (Figure 2.13).

The main idea of GAN is to perform an adversarial two-player min-max game between a Generator and a Discriminator. The objective of the Discriminator is to maximize the distance between real image and Generator image proving generated image as fake. On the contrary, the objective of Generator is to generate image that looks like real i.e. minimizes the distance between real and generated image. This min-max game continues to achieve equilibrium so that Discriminator is expert in identifying fake images and Generator is expert in producing highly realistic images. The Generator takes as an input a random noise, and produces a fake image. Discriminator takes as input an image and predict if this image is real or generated.

Following the training strategy of GAN, very recently GAN-like adversarial training has been explored in node embedding [Ding et al., 2018, Wang et al., 2019] and knowledge graph embedding [Wang et al., 2017a, Cai and Wang, 2018]. Training GAN for graph embedding is a complex process due to the fact that it can not be used to generate discrete samples like natural language sentences or knowledge graph triples because the discrete sampling step prevents gradients from propagating back to the Generator [Cai and Wang, 2018]. IRGAN [Wang et al., 2017a], KBGAN [Cai and Wang, 2018], KSGAN [Hu et al., 2019] have successfully trained Generator for discrete sampling borrowing concepts from reinforcement learning. To be precise, authors used policy gradient in its simplest form to train the Generator. In the proposed framework, we used a form of policy gradient to train the Generator with reward coming from the Discriminator.

In the proposed framework (Figure 6.1), we adopt a training framework that closely follows the framework proposed in GraphGAN [Wang et al., 2019]. GraphGAN [Wang et al., 2019] is designed to work for node embedding in homogeneous network. Therefore, it does not learn any representation for the relation types connecting the various types of nodes in the knowledge

graph. In GraphGAN, the generator is trained to select the neighbor nodes that are probable true neighbors. It uses a breadth-first-search strategy to select negative samples to train the Discriminator. GraphGAN is proposed for homogeneous graphs. Here, we propose a similar adversarial training framework that works for knowledge graphs, and the model is designed specifically for automatic protein function annotation. We have redesigned the Discriminator to work on triples instead of node-neighbor pairs. The Generator is formulated to guide a random walk over the knowledge graph. Like an agent in reinforcement learning, Generator trains the random walk to discover hidden link between protein and GO annotations by updating its entity and relation embeddings, and by manipulating rewards from the Discriminator. During the training, at each iteration, Generator is updated to produce more realistic triples given source entity and relation type by optimizing itself using rewards from the Discriminator. From this point of view, we retain the original goal of GAN where Generator actually learns to produce real-like samples. The existing GAN-based knowledge graph embedding approaches, for example, KBGAN [Cai and Wang, 2018] use GAN to produce negative triples. The negative triples are then used to train TransE-like KGE models. Although this is an important step for training a effective KGE model, GAN has potential to produce unseen positive triples from the knowledge graph. In Prot-A-GAN, we explore the potential of GAN to discover hidden links from the knowledge graph to perform automatic protein function annotation task.

6.2 Generator and Discriminator Models

In the proposed Prot-A-GAN framework, given a knowledge graph G, our target is to learn two models : the Generator, and the Discriminator by training them adversely. In the following sections, we describe Generator and Discriminator models separately.

6.2.1 Generator Model

The **Generator** model tries to learn connectivity distribution as $prob(o|s,p)$ for triple (s,p,o). The Generator is associated with a random walk, and it works as follows :

- It starts with a random initialization of its parameters : 1) entity embeddings Θ_G^V and 2) relation embeddings Θ_G^R
- The Generator guides a depth-limited target-specific random walk that takes as input a set of subject entities and a set of target entities of certain type, and a relation r, from which we know beforehand that valid triples can be formed out of the two sets using relation r. For each subject entity v_i, Generator performs a depth-limited random walk. It stops as soon as it reaches an entity v_j for which $\phi(v_j)$ is the type of the target entities. In other words, random-walk stops when it reaches one of the target entities. In case it can not reach a target entity, it stops as soon as it reaches the depth limit.
- The target entity and subject entity together with the prescribed relation form a generated triple (v_i, r, v_j). A number S of such triples are generated per subject entity.
- All of these generated samples are sent back to the Discriminator to judge, and the rewards are returned back to the Generator, telling how negative the samples were. These rewards are then used to update the Generator parameters i.e. Θ_G^V and Θ_G^R so that the Generator produces more positive-likely triples.

The Generator optimizes the following loss : For a triple (s,p,o), Discriminator scoring function $f_d(s,p,o)$, and Generator scoring function $f_g(s,p,o)$, we compute the reward as,

$$reward = log(1 + \exp f_d(s,p,o))$$

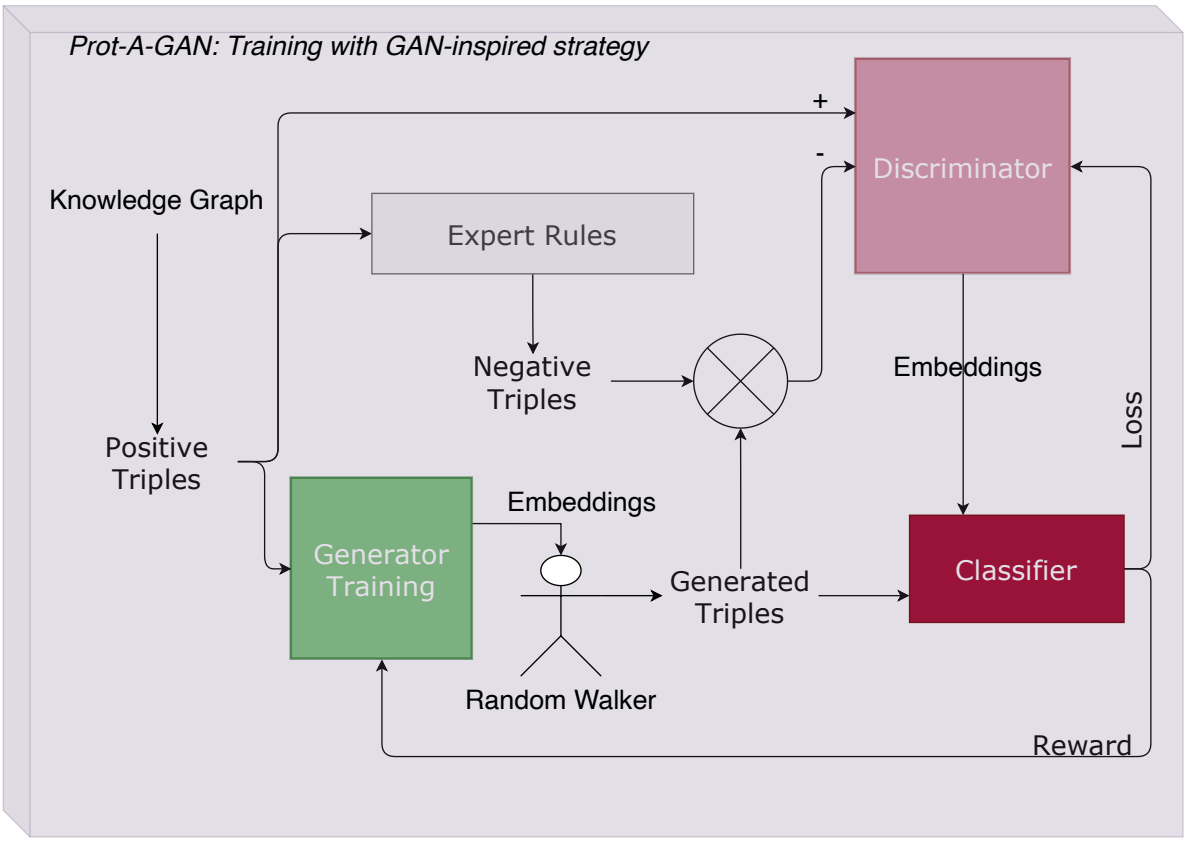


FIGURE 6.1 – Schematic diagram of the proposed Prot-A-GAN framework (Training). The framework has two main components, Generator and Discriminator. Generator is attached with a random-walker that discovers new triples from the knowledge graph. It uses embeddings learned by Generator to compute relevancy score for selecting the path. The triples generated by random-walker are feed to the Discriminator along with negative triples produced by applying expert rules. The Discriminator is trained with positive triples which are the true facts in the knowledge graph, and the negative triples from expert rules, and Generator. The Discriminator and Generator are trained in a adversarial manner to learn two sets of embeddings : 1) Θ_G^V and Θ_G^R for Generator, and 2) Θ_D^V and Θ_D^R for Discriminator. During the training the model is trained for certain number of epochs. In each epoch, the Discriminator and the Generator are individually run for another certain number of iterations. The annotation procedure is shown in Figure 6.2

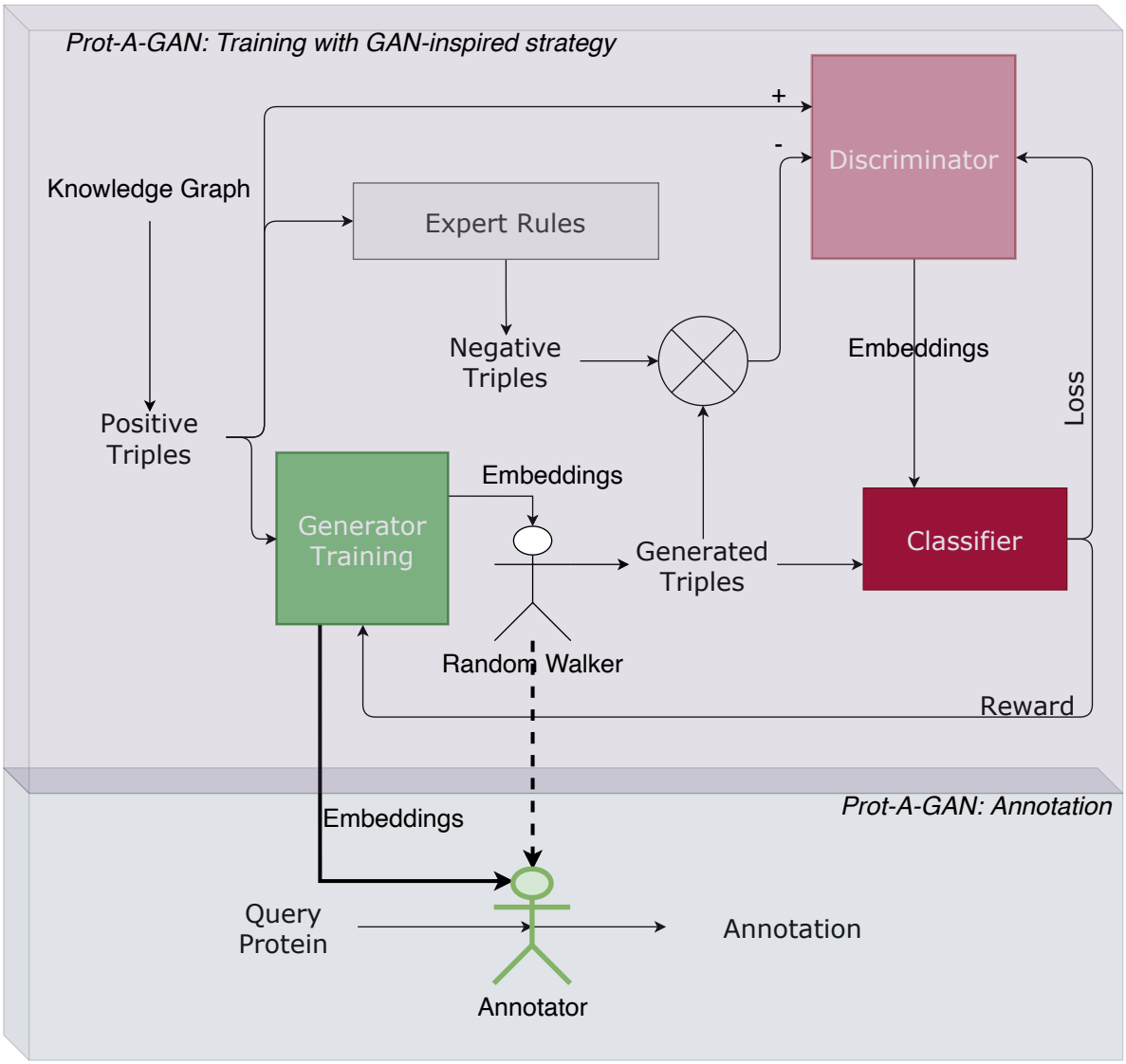


FIGURE 6.2 – Schematic diagram of the proposed Prot-A-GAN framework (Annotation). Once the model is trained following the Figure 6.1 , the Generator parameters i.e. Θ_G^V and Θ_G^R are used to guide the random-walker (now annotator) to perform the annotation task. It takes a protein ID as input, the annotator moves around the knowledge graph to find relevant GO terms using the relevancy score computed from Θ_G^V and Θ_G^R.

, and the loss function become as follows (Equation 6.1) :

$$L_G = log(\sigma(f_g(s, p, o)) \times reward) \tag{6.1}$$

Generator starts with a random initialization of its parameters. The parameters are then updated by optimizing the loss function shown in Equation 6.1. The training of the Generator follows : 1) a random walk that generates triples, 2) a Discriminator providing feedback in the form of rewards, and 3) an optimizer optimizes the loss. Preparation of training data follows the process described in Algorithm 2. The Algorithm 2 also serves the purpose of generating annotations once the Generator is well trained. This algorithm is a very important part of the Prot-A-GAN framework as the success of the Generator is largely depends on the training data generated using the algorithm. The random walk involved in the process is the core addition to the approach as it provides a mean to connect proteins with remote annotation by walking through the knowledge graph guided by the Generator parameters i.e. entity embeddings (Θ_G^V) and relation embeddings (Θ_G^R). In the training, We have used L2 regularization.

Algorithm 2: Training data preparation for Generator

Input: Θ_G^V : entity embeddings, Θ_G^R : relation embeddings, $\mathcal{A}_1$: Source_entity_type, $\mathcal{A}_2$: Target_entity_type, r : relation, d : depth limit
Output: T_g : Generated triples

```
A1= "protein"
A2= "GO"
Tg ← {}
for vi ∈ V and φ(vi) == A1 do
    for 1 to d do
        Perform one step of depth-limited random walk and get next entity vj
        if φ(vj) == A2 then
            Tg = Tg ∪ {(vi, r, vj)}
        else
            Continue random walk with vj as subject entity
```

In knowledge graph embedding models scoring function plays an important role. Scoring functions, normally, gives a numerical assessment regarding the closeness of the entities in a triple based on the relations they hold. In Prot-A-GAN, we use a scoring function based on translational distance - a commonly used scoring function. For the both Generator and Discriminator, we have used the same scoring function (for Discriminator Equation 6.3 and for Generator Equation 6.2) :

$$f_g(s, p, o) = -||s + p - o||_2 \tag{6.2}$$

Similarly,

$$f_d(s, p, o) = -||s + p - o||_2 \tag{6.3}$$

These scores are computed using entity and relation embeddings from respective models i.e. Generator or Discriminator. The same symbols s, p and o are interchangeably used to represent both discrete entities and numerical representation of the entities.

The proposed target-specific depth-limited random walk works by following the direction guided by the scoring function and by Generator parameters namely Θ_G^V and Θ_G^R. Given a subject entity, the random walk moves to the next entity by probabilistically choosing the path according to scores of the neighbors. The scores are transformed into probabilistic scores by

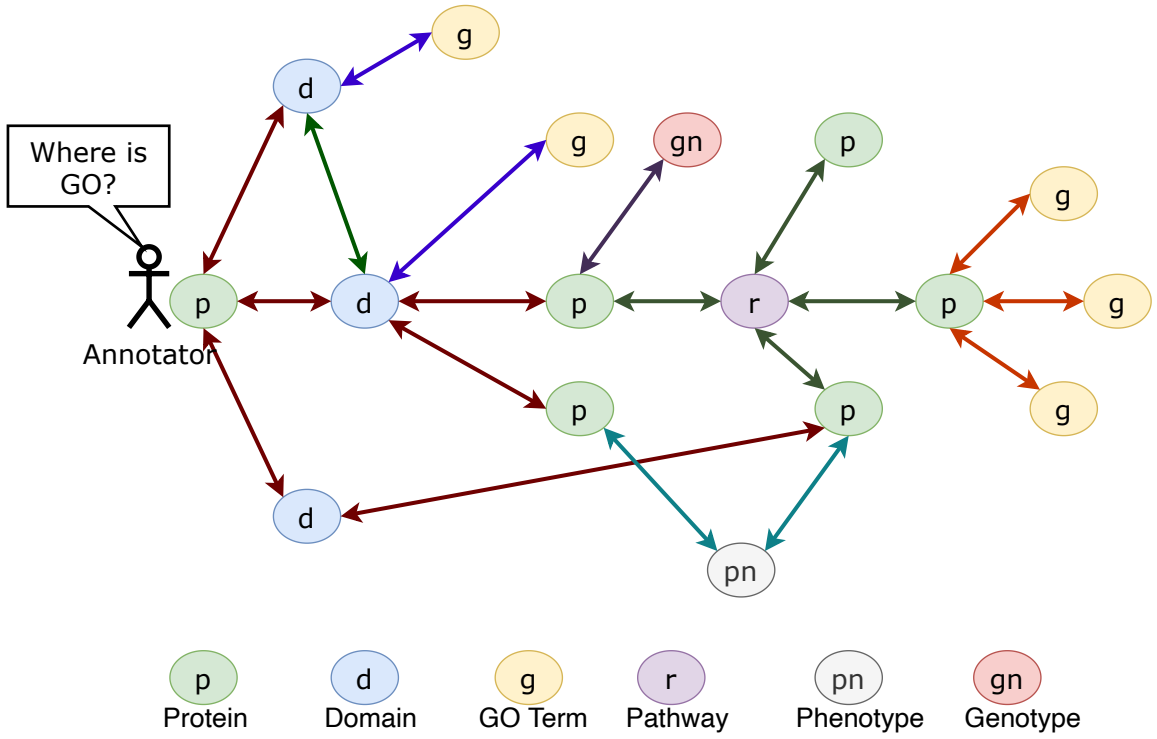


FIGURE 6.3 – Depth-limited target specific random walk. The random walker begins at a query protein. It looks through the different directly connected edges to explore the different path directions in search for GO annotations.

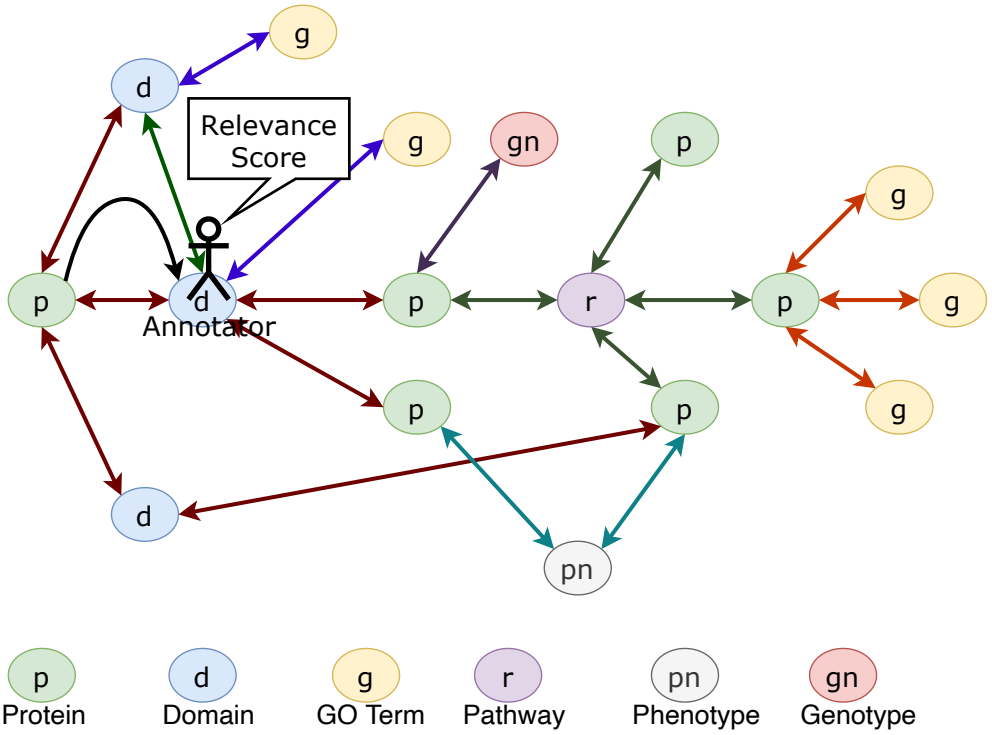


FIGURE 6.4 – Depth-limited target specific random walk. The walker moves to the next step based on relevance score computed from Generator parameters.

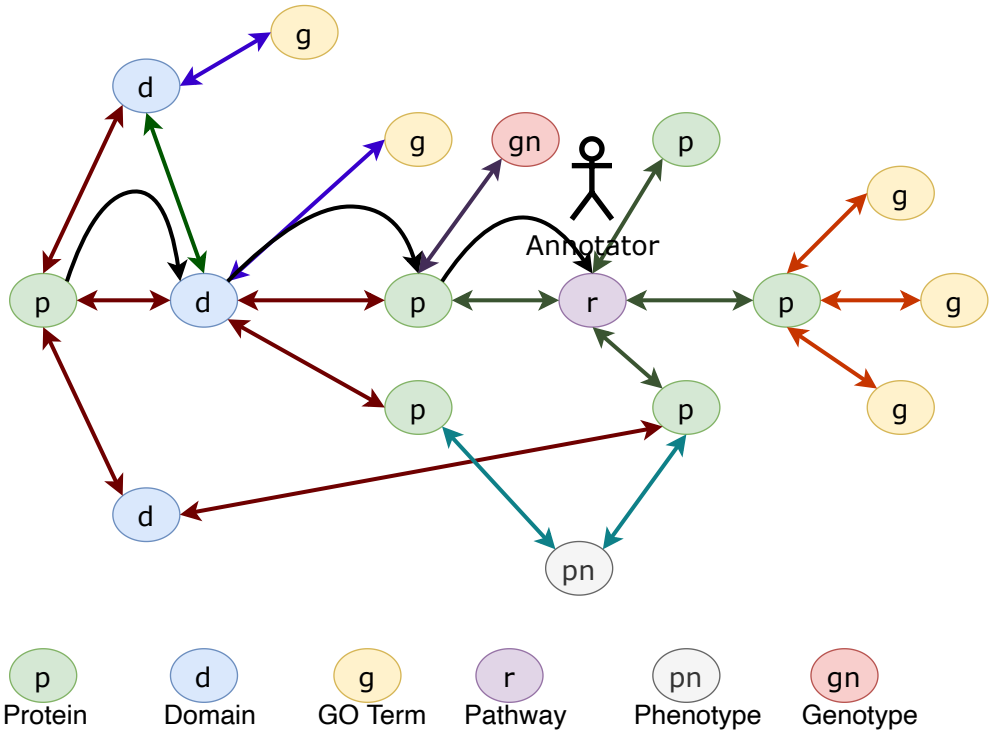


FIGURE 6.5 – Depth-limited target specific random walk. The walker traverses the knowledge graph following the Generator-guidance until it gets to annotation terms.

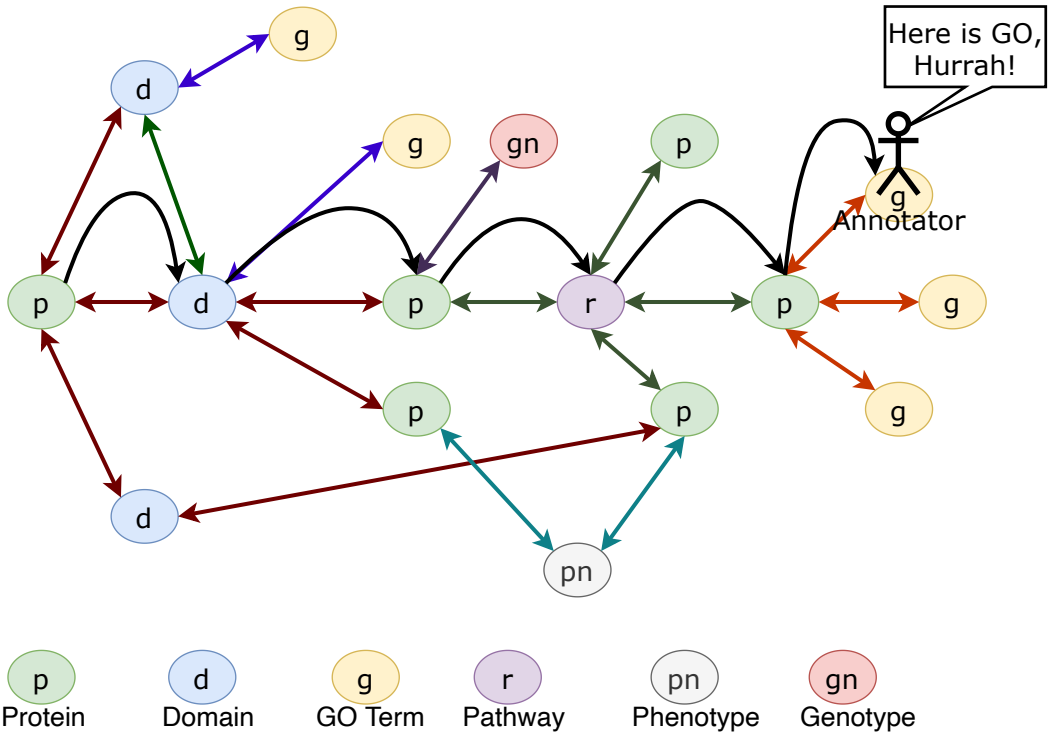


FIGURE 6.6 – Depth-limited target specific random walk. The walker stops as soon as reaches to an annotation term i.e. GO terms.

using softmax function over all of the directly connected entities. Random walk keeps moving until it reaches one of the target entities or it reaches the predefined depth along the path. This strategy greatly reduces the computational costs of finding the target entities and the random walk does not run forever if it can not reach a target entity. The random selection of the next node is governed by the relevance score of the triples. The triples made of the subject entity, the prescribed relation type and the selected target entity added to the set of training samples for Generator. Moreover, the selected target entities complete the missing links in the knowledge graph once the Generator is well-trained. To put the importance of the proposed random walk in the context of function annotation task, let us consider we want to find the GO annotations for a protein. The random walk starts from the protein, moves along the path chosen by probabilistic relevance score and ends up selecting entities that are GO terms. Each random walk selects one GO term. Random walk is run several times with the same subject protein to find multiple GO annotations. The steps of the random walk are depicted in Figure 6.3, 6.4, 6.5, and 6.6.

Let us consider a set of triples made up of directly connected neighbor entities, and X be the list of scores computed using Equation 6.2 for the triples. The relevancy score is computed as a softmax probability over X (Equation 6.4) and finally the relevancy score is used as probability in random selection of the next node in the random walk.

$$relevance - score = \frac{e^{X-max(X)}}{\sum e^{X-max(X)}} \tag{6.4}$$

6.2.2 Discriminator Model

Discriminator model learns to associate triples with a likelihood score that probabilistically classify them into positive or negative triples. Depending on the scoring function the optimization can be maximization or minimization problem. In the proposed model, we adopted translational distance as the scoring function for Discriminator that is shown in Equation 6.3. That means, the Discriminator gives high score for negative triples and low score for positive triples.

To train the Discriminator model parameters, namely Θ_D^V and Θ_D^R, we take the sigmoid of the scores and minimize the following binary cross entropy loss :

$$L_D^+ = \sum max(f_d^+(s,p,o),0) - f_d^+(s,p,o) * z^+ + log(1 + exp(-abs(f_d^+(s,p,o)))) \tag{6.5}$$

$$L_D^- = \sum max(f_d^-(s,p,o),0) - f_d^-(s,p,o) * z^- + log(1 + exp(-abs(f_d^-(s,p,o)))) \tag{6.6}$$

$$L_D = L_D^+ + L_D^- \tag{6.7}$$

Here, $f_d^+(\cdot)$ is score for positive triple and $f_d^-(\cdot)$ is score for negative triple. z^+ and z^- denote the positive and negative labels, usually 1 and 0 respectively. From the Equations 6.5, 6.6, and 6.7, it is evident that training Discriminator with the loss L_D requires to sample negative triples. We use random walk guided by Generator parameters (Θ_G^V,Θ_G^R) to sample negative triples probabilistically and knowledge of protein biology to sample strictly negative triples from the knowledge graph.

Sampling Negative Triples for Training Discriminator

Preparing negative data is vital to efficiently train a Discriminator. Followings are the steps we take to sample the negative facts from the knowledge graph (also shown in Algorithm 3).

— For a particular relation type p, we collect all the nodes acting as tail in the triples. Let us call this set as U.
— For a particular entity s, and for the same particular relation type p as above, we find the directly connected tail-nodes. Let's call this set T_s.
— After that, we compute the set difference $U \setminus T_s$ to find the candidates for forming negative facts. For a particular entity s and relation p, the item $o \in T_s$ serves as the tail nodes when building the triples of the form (s, p, o).
— For each candidate triple, we compute the relevancy score using Generator parameters Θ_G^V, Θ_G^R and rank them based on this score. The least relevant triples are popped up as most negative triple. We select one negative triple against one positive triple. The relevancy score is computed as a softmax probability over all the the candidate facts. Let's consider, x is the score for a particular triple computed using Equation 6.3 and X is the list of the scores for all of the candidate triples, the relevancy score computed using the Equation 6.4

Algorithm 3: Negative Sampling for Discriminator

Input: Θ_G^V, Θ_G^R, v_i : subject entity, r : relation , S : sample size
Output: $T^- = []$
$U = \mathcal{N}^r$; $\mathcal{N}^r$ contains all tail nodes for a relation r
$T_s = \mathcal{N}_{v_i}^r$, the directly connected entities of v_i following the relation r
$T = U \setminus T_s$
$E = \{(v_i, r, t) \forall t \in T\}$ set of negative triples
$X = f_g(x)\ \forall x \in E$, holds the scores for all of the candidate triples in T using Θ_G^V, Θ_G^R
relevance-score$=\frac{e^{X-max(X)}}{\sum e^{X-max(X)}}$
E^- = randomly sample one triples from E according to relevance-score to match against the positive triple

6.2.3 Training Algorithm

Here, we present the combined training strategy for the Prot-A-GAN. The training is run for a pre-defined number of iterations. In each iteration, the Discriminator is trained followed by the training of the Generator. In each iteration, updated training data is prepared for Discriminator by sampling negative triples using the Algorithm 3, and the parameters are updated using the loss function. Similarly, Generator generates updated set of facts following the Algorithm 2. Apart from this, both Discriminator and Generator updates the training data after a certain interval. Based on the rewards provided by the Discriminator, the Generator parameters are updated.

Algorithm 4: Training of ProtAGAN

Input: Knowledge graph G, Number of training epochs n_epochs, Discriminator Interval $interval\_dis$, Generator Interval $interval\_gen$
Output: Θ_G^V, Θ_G^R, Θ_D^V, Θ_D^R

```
Initialize the model parameters Θ_G^V, Θ_G^R, Θ_D^V, Θ_D^R
for epoch <= n_epochs do
    for d_epoch <= n_epochs_dis do
        if d_epoch reaches interval_dis then
            get_train_data_for_dis(G)
        Compute the Discriminator loss
        Update the Discriminator parameters Θ_D^V, Θ_D^R
    for g_epoch <= n_epochs_gen do
        if g_epoch reaches interval_gen then
            get_train_data_for_gen(G)
        Compute the reward for the Generator using the Discriminator parameters Θ_D^V, Θ_D^R
        Compute the loss for the Generator
        Update the Generator parameters Θ_G^V, Θ_G^R
    epoch ← epoch + 1
```

6.3 Experiments and Results

To evaluate our proposed framework, we test its performance on a protein knowledge graph for the link prediction task that corresponds to automatic function prediction. we have designed and built a knowledge graph supporting protein function annotation. The construction of the knowledge graph is explained in the following section 6.3.1

6.3.1 Construction of UniProtinKG knowledge graph

We have designed Prot-A-GAN specifically for automatic function prediction. Therefore, we have built a knowledge graph by curating protein information from UniProtKB/SwissProt database. The edge-lists are prepared semi-automatically as the relation types are hand-crafted. We are interested in leveraging the huge amount of curated information available in UniProt/SwissProt for discovering new knowledge by building a knowledge graph "UniProtinKG" : UniprotKB in Knowledge Graph. The main objective of UniProtinKG is to integrate annotation data from SwissProt and domain information from TrEMBL in a knowledge graph format and to manipulate this knowledge for further knowledge discovery. UniProtinKG also takes care of biological ontologies, for example, it associates hierarchical relationships from GO ontology, InterPro signatures, Reactome ontology that eventually help in finding remote connections while exploring the knowledge graph.

Although reviewed and annotated proteins are associated with numerous information, integrating all of them into a single knowledge graph will make it prohibitively large for the downstream analysis such as training embedding models and reasoning over it. GO annotation data is the must-have property for reviewed annotated proteins. Along with GO annotations, we have carefully selected information that are related to the protein functions, for example, when available, we include pathways, domains, genotype, phenotype, tissue, glycosylation data

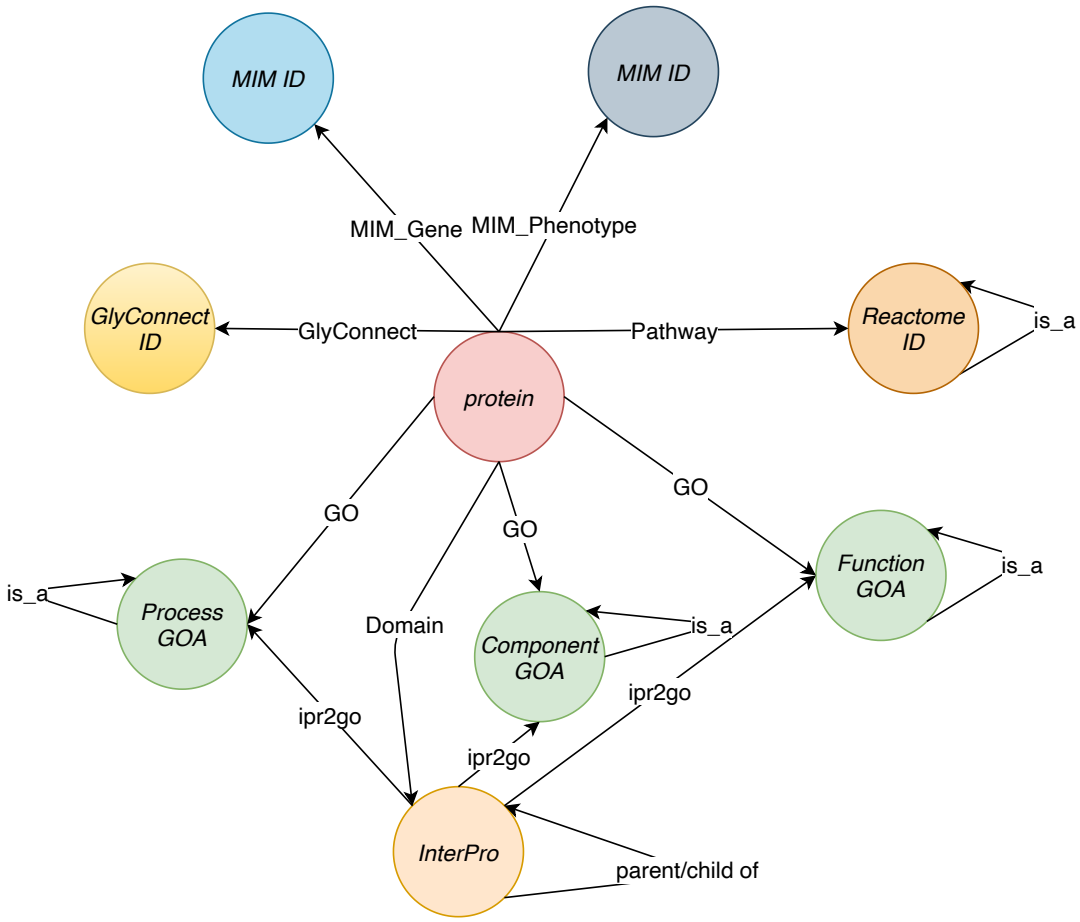


FIGURE 6.7 – Schema of the protein knowledge graph

for annotated proteins. In the case of un-annotated proteins, we only include InterPro signatures which is available from the sequence data. To build the "UniProtinKG", we have designed a parser that process UniprotKB data coming in text format, collect the pre-mentioned attributes and integrated into a knowledge graph. In this KG, nodes are heterogeneous, i.e. nodes are of different types : we have proteins, domains, GO terms, Reactome terms, OMIM terms etc. While deciding the edge, we have hand-engineered the edge type to reflect the nodes and the relations among them.

Among many information types, we include the following protein attributes from UniProtKB to build the "UniProtinKG" graph :

— Protein ID : Each protein in UniProtKB is identified with an entry identification code. When we say a protein, we point to a specific UniprotKB ID that references this protein in the knowledge graph.

— Domain : Domains are the evolutionary information conserved in protein sequences. There are many computational tools that identify the domain information from protein sequence e.g. Pfam, gene3d, Prosite etc. However, InterPro [Mitchell et al., 2018a] is an integrative database that uses existing tools to identify the domains and give them unique signatures. We have used InterPro signatures from InterPro database.

— GO annotation : Manual annotation of protein is the direct assignment of Gene Ontology (GO) terms to proteins, ncRNA and protein complexes by curators from evidence extrac-

ted during the review of published scientific literature, with an appropriate evidence code assigned to give an assessment of the strength of the evidence. We have only considered the annotations that have evidence code other than IEA. IEA indicate electronic or automatic annotations. GO terms are categorized into 3 aspects : 1) Biological Process, 2) Molecular function and 3) Cellular component. Based on these three aspects, GOA annotations are linked with proteins using three relation types "GOA". GO terms are linked as a Directed Acyclic Graph (DAG). As the ancestral relations are important, we have used the "is_a" relation that connects a GO term with its parent GO terms. Interestingly, we also include GO annotations of InterPro domains, obtained through the ipr2go resource [Mitchell et al., 2018a].

— Pathway : Reactome is a freely available, open-source relational database of signaling and metabolic molecules and their relations organized into biological pathways and processes. The core unit of the Reactome data model is the reaction. Entities (nucleic acids, proteins, complexes, vaccines, anti-cancer therapeutics and small molecules) participating in reactions form a network of biological interactions and are grouped into pathways. Pathway information of proteins is included using Reactome database reference ids.

— Glycon-protein binding : Glycosylation defines the adhesive properties of proteins and cells. The immune system largely functions via glycan-protein interactions, which highlights the importance of glycans in physiology, pathogen recognition, cancer and autoimmune diseases. GlyConnect is a platform integrating sources of information to help characterise the molecular components of protein glycosylation. The primary focus of GlyConnect is on the relationships between glycans and proteins that bear them. Glycan molecules modulate many other processes important for cell and tissue differentiation, metabolic and gene regulation, protein activity, protein clearance, transport and more. Glycon-protein binding is an important information that is related to the functional behavior of proteins.

— Gene and genetic phenotypes : OMIM is a comprehensive, authoritative database of human genes and genetic phenotypes. OMIM cross-references are used to map genes and genetic phenotypes to proteins. We include OMIM cross-reference ids into the knowledge graph that helps in grouping similar types of proteins.

The schema of the UniProtinKG is shown in the Figure 6.7. It shows how a protein is connected with its attributes. When many proteins are put together following this schema, the knowledge graph is formed from the connection to common attributes. This eventually opens up possibility of discovering remote links among the entities.

6.3.2 Implementation

Prot-A-GAN is implemented using Python and Tensorflow machine learning framework similar to the implementation of GraphGAN. The training stage does not have any pre-training step. It starts from initializing the embedding matrices from a random distribution. The Discriminator loss function described in Equations 6.5, 6.6, and 6.7 is realized using *sigmoid_cross_entropy_with_logits* functions provided in Tensorflow.

6.3.3 Evaluation Protocol

At first, we report the evaluation strategy for measuring the relation prediction task. In this task, given two nodes, we want to know which is the appropriate relations that might best connect them. From a set of test triples, given head and tail nodes, we score all the triples formed

by placing any type of relation using the Generator embedding parameters. And then we rank the relations based on their scores and select the top-ranked relations. The relation-prediction accuracy is measured by following formula similar to Hit@1 i.e. we record the top ranked relation type and match it with the true relation type.

$$Hit@1 = \frac{1}{|\mathcal{E}_t|} \sum_{(s,p,o)\in\mathcal{E}_t} I(rank_p = 1)$$

where $\mathcal{E}_t$ is a set of test triples in relation prediction task, I$(\cdot)$ is an indicator function representing whether the rank of the true relation among all other predicted relations is 1 or not.

Then, following the previous works on protein function annotation, we adopted precision, recall and F1 measure to evaluate the performance of the proposed framework in protein annotation task. To proceed with evaluation, we first prepared a set of proteins as test proteins. To be rigorous, we assume that the test proteins contain only domain information from InterPro database. This means that each test protein is attached to the UniProtinKG graph by their domain composition through *protein : has_ domain :ipr* relation. And, for the training proteins, we have more attributes than only domains, for example, GO terms, pathway, OMIM etc.

The knowledge graph comprises of both training proteins and test proteins. After training the model using the framework described in Figure 2.13, we run the random walk for each test protein. The random walk searches for GO annotations following the guide of the trained Generator, precisely using Θ_G^V and Θ_G^R. Let us consider that P is the set of predicted GO terms and A is the set of ground-truth GO terms for a particular test protein q. We provide precision, recall and F1 measures as shown in the following formulas :

$$precision = \frac{|A \cap P|}{|P|},$$

$$recall = \frac{|A \cap P|}{|A|},$$

$$F1 - measure = \frac{2 \times precision \times recall}{precision + recall}.$$

The final measure includes average precision, average recall and average F1 measure.

6.3.4 Results

In this section, we present the precision, recall and f-measure for function annotation task on UniProteinKG. As a proof of concept, the knowledge graph is built on the disease proteins from UniProtKB. We call this as UniProteinKG-Disease graph.

The UniProteInKG-Disease KG contains total number of entities, $|\mathcal{V}| = 26755$, total number of edges $|\mathcal{E}| = 321596$, $|\mathcal{R}| = 18$ different types of relations, and $|\mathcal{A}| = 8$ types of entities built from UniProtKB/SwissProt disease proteins.

To assess the performance of Prot-A-GAN, we report the relation prediction accuracy. For each triple (s,p,o) in the test set, for each r in the graph, we replace p by r and measure the score. We pick the top scorer and look for the relation. If the relation comes out to be p, we take it as a correct prediction.

In this task, we kept the dimension of the embeddings to 150. During the adversarial training stage, we train for 50 epochs, with mini-batches of training samples of size 1024 for each epoch.

Embedding	#Test triples	#Correct predictions	Accuracy
Random	17676	38	0.002
Discriminator	17676	11953	0.676
Generator	17676	16208	**0.917**

Table 6.1 – Performance of Prot-A-GAN for relation prediction using UniProteinKG-Disease

We use the self-adaptive optimization method Adam for all training and always used the default settings recommended by Tensorflow. Inside the main training loop, Discriminator is trained for 30 epochs and after 15 epochs, it gets new samples of negative edges. Similarly, Generator is also trained for 30 epochs for each main epoch. And after 15 consecutive epochs, it generates a new set of edges to collect reward and update its parameters accordingly.

The test triples only contains the edges linking proteins and GO terms. As the Generator is designed to discover links between proteins and GO terms, intuitively, the task of relation prediction between protein and GO term should be better served by Generator embeddings. The result presented in the Table 6.1 confirms the intuition by having the top accuracy using Generator embeddings.

The problem of functional annotation of protein tries to find the right association between GO terms and protein. That means, for a given protein, we are interested to find the GO terms that describe the functions it might be performing in the body. To check the performance of Prot-A-GAN in function annotation task, we have separated a list proteins as test proteins. For these proteins, we only have InterPro domain composition in the knowledge graph. They do not have any kind of annotation. The task is to find the annotations using the Prot-A-GAN approach. The ground-truth annotations for the test proteins are known apriori to assist in computing the precision, recall and f1-measure.

For this task, we again kept the dimension of the embeddings to 150. However, during the adversarial training stage, we train Prot-A-GAN for 30 epochs, with mini-batches of training samples of size 512 for each epoch. We use the self-adaptive optimization method Adam for all training and always used the default settings recommended by Tensorflow. Inside the main training loop, Discriminator is trained for 5 epochs. Similarly, Generator is also trained for 5 epochs for each main epoch.

Once the model is trained, the Generator parameters are used to guide the random walk or the annotator to find the GO annotations. For each query protein, the annotator discovers GO annotations. GO terms are organized in the Gene Ontology in a hierarchical fashion from generalized to specialized. For each of the predicted GO terms we collect the ancestor terms. We do the same for the terms in ground-truth set. After that we compute the precision, recall and F1-measure for each protein. Table 6.2 shows the outcome of the experiment. The precision, recall and F1-measure :

In Table 6.2, we present the precision, recall and F1 score for 799 test proteins. These 799 proteins were separated during the pre-processing step. We do not include information other than domains for these test proteins. For each protein in the test-set, we run Prot-A-GAN annotator and record the predicted GO terms.

We see that Prot-A-GAN has better precision when we run the annotator for only one time. In this case, the Prot-A-GAN run random walk as many times as the number of domains associated with the query protein. As each domain gives a new path leading to a GO term if found, the number of predicted GO terms is equal to the number of domains or less if Prot-A-GAN fails to reach a GO term. As we increase the number of run, we see a gradual decrease in the precision

TABLE 6.2 – Automatic protein function annotation using Prot-A-GAN on UniProteinKG-Disease

#RUN	POST-PROCESSING	PRECISION	RECALL	F1-MAX	#ANNOTATED
1	YES	**0.609**	0.190	0.255	746/799
	NO	0.325	0.074	0.108	746/799
2	YES	0.587	0.281	0.331	764/799
	NO	0.330	0.124	0.156	764/799
5	YES	0.498	0.394	0.376	765/799
	NO	0.302	0.204	0.199	746/799
10	YES	0.415	**0.499**	0.376	**765/799**
	NO	0.257	0.281	0.207	765/799

and a gradual increase in the recall. This behavior is explainable from the fact that when the number of run increases, the number of predicted GO terms increases. This increases the number of false positives. Thus a reduced precision is observed. However, the high precision for single run indicates that Prot-A-GAN has the potential to discover high quality annotations. At the same time, the low recall indicates the Prot-A-GAN can not discover all of the annotations. In the case of 10-run, we see a high recall compared to 1-run, 2-run and 5-run as it discovers higher number of GO terms by running for 10 times. Intuitively, the high number of GO terms introduce high number of false positives leading to a reduced precision.

6.4 Conclusion

To the best of our knowledge, this is the first time GAN is merged with knowledge graph embedding to design an automatic protein function annotator. The objective was to build a machine learning pipeline that leverage the power of adversarial learning on knowledge graph to discover functional annotations of proteins. The proposed approach opens a new direction in the research of automatic protein function annotation leveraging the power of GAN and knowledge graphs. Functional annotation using GO is relatively difficult. GO terms are arranged hierarchically in Gene Ontology. Moreover, each protein is annotated with multiple GO terms that can be distinctly placed in the ontology. Sometimes, a single protein can have more than 30 GO terms. Therefore, deciding on the number of annotations to generate is a dynamic decision. However, we varied the number of predicted annotations by varying the number of run. This is one of the reason we have lower recall than precision when the predicted set is smaller than the ground-truth set. Apparently, the precision seems to be lower before the post-processing is performed. However, applying post-processing approaches to gather around the ancestor annotations improved the outcome significantly.

Training adversarial models are very resource intensive. Applying adversarial training on knowledge graph is computationally very expensive and requires advanced hardware settings. The hyper-parameters involved in the process have significant impact on the outcome of the experiment. Finding the right hyper-parameter configuration is very challenging due to the large search space and resource intensive training. Due to time and resource constraints, we could not present a thorough analysis of the impact of hyper-parameters on the experimental outcome. However, after few trials, we found this setting to be promising. This is a proof-of-concept and results are preliminary yet promising. Further experimentation is necessary to justify the performance.

While we do not have detail annotations for most of the proteins sequenced to-date, there are few hundred thousands of proteins that are manually reviewed, annotated with large body of information ranging from 3D-structure to pathways, diseases and drug targets. Furthermore, the information is coming from different data sources generated using various biological experiments performed across the globe. This is a huge challenge to integrate these many sources. And at the same time, it provides opportunity for efficient computational protein function annotation. In the knowledge graph, we integrate many data sources related to proteins, pathways, genotype-phenotype, functions etc. Along with many other types of entities, once we have protein and their functions as the entities of knowledge graph, we can translate the task of protein function annotation into a link prediction problem over a protein knowledge graph. The basic idea is to learn vector representation of proteins and functions using Prot-A-GAN. After that, given a query protein, link prediction is performed between query protein and functions and a fixed number of top-ranked functions are listed as predicted functions. In summary, in this work, we present a knowledge graph embedding approach targeting the application of automatic protein function annotation and leveraging the power of GAN.

Quatrième partie

Epilogue

7

Conclusion and Future Work

7.1 Conclusion

Automatic functional annotation of proteins is an open research problem in bioinformatics. Decades of research have been dedicated to solve this challenging problem. Due to the proliferation of sequence data, thanks to advanced sequencing technology, we have now millions of data points that are available in public databases. The growing number of protein entries in public databases, for example in UniProtKB, poses challenge in manual functional annotation. Manual annotation requires expert human curators to search and read related research articles, interpret the results, and assign the annotations to the proteins. Thus, manual annotation is time consuming and expensive. Although manual annotation is the most accurate way of functional annotation, it is not a feasible way to keep pace with the asymptotically increasing amount of protein entries accumulating in the databases. Therefore, designing computational tools to perform automatic annotation leveraging the high quality manual annotations that already exist in UniProtKB/SwissProt is an important research problem.

To contribute to the research of automatic protein function annotation, the central objective of this thesis was to develop computational approaches for functional annotation of proteins. Earlier researches have explored various protein attributes such as sequences, structures, domains, protein-protein interaction network, physico-chemical properties, along with various computational approaches such as association rule mining, machine learning, deep learning, natural language processing, representation learning, network analysis etc. In this thesis, we explored graph-based approaches for functional annotation of proteins. Through experimental analysis, we observed the strong association of domains to the functions, particularly in case of EC annotation.

In chapter 3, we introduced GrAPFI (Graph-based Automatic Protein Function Inference). GrAPFI is an EC annotation technique that explores network of proteins for functional inference. Instead of protein-protein interaction network, the network is built of domain similarity where nodes are proteins and edges represent similarity in domain composition. In-depth experimental analysis shows promising outcome in EC annotation. GrAPFI is a simple explainable neighborhood-based annotation approach. GrAPFI results are easily explainable as it employs interpretable scoring function. GrAPFI has certain limitations. There are many short-length protein sequences for which InterPro domains are not available. In such cases, GrAPFI is unable to find annotation as the proteins can not be linked to the underlying protein-network.

Another limitation of GrAPFI is that in its original form it can not perform GO annotation. In chapter 4, we extended GrAPFI to perform GO annotation. To improve the poor performance in GO annotation, we proposed a post-processing pipeline that significantly improved the raw

performance. We also showed that the proposed post-processing framework can be used with other off-the-shelf annotation tools to improve the precision. The proposed framework works by finding the membership score to filter out the GO terms that are weakly linked to the group. Although it improves the precision, the recall is declined due to the fact that it predicts less GO terms for each protein.

Learning representation also known as embedding is an important research area in machine learning that aims to find meaningful vector representation of real world entities. To take advantage of the recent advancement in representation learning in natural language processing, in Chapter 5, we applied word embedding techniques to embed InterPro domains in low dimensional vector space and using those embeddings, in later stage, we annotated proteins with EC numbers. This approach shows significant improvement in the accuracy of EC annotation over 3-mer based embedding. One of the disadvantage of using this method is that it does not provide any explanation of the prediction as it employs complex neural network to learn low dimensional vectors of protein domains. This chapter is significant because it introduces word2vec word embedding models (Skip-gram and CBOW) that form the basis for a wide range of representation learning techniques for graph structured data, such as node2vec, deepwalk, RDF2vec etc. Recently a group from Heidelberg re-used our idea of protein domain embedding in a Dom2Vec approach [Melidis and Nejdl, 2021].

One of the drawbacks of the previous approaches is that they work on protein-domains only. However, there are many other attributes such as pathway, genotype, phenotype, taxon that might incorporate important insight in the process. Moreover, hierarchical relations among the attributes represented as ontology can be meaningful addition to the process. In chapter 6, we explored knowledge graph to integrate various data sources and employ it for protein function annotation. We propose Prot-A-GAN, a knowledge graph embedding technique using GAN-inpired adversarial learning. We trained a generator that using a random walk finds GO annotations for proteins. To be rigorous and coherent with the previous approaches, the test proteins are only accompanied with InterPro domains while performing random walk to select GO annotations. As it is observed in the experiment, Prot-A-GAN shows promising outcome in relation prediction as well as in GO prediction. Although still a proof-of-concept method, it shows promises in relation prediction and function prediction alike. There is still scope of improvement in GO annotation process by improving the random walk and finding appropriate scoring function. Prot-A-GAN requires further theoretical development for better understanding of the method as well as finding right configuration to achieve equilibrium in adversarial learning.

7.2 Future Research Directions

In this thesis, we explored graph-based approaches for automatic protein function annotation. Graph is a rich area of research in computer science with centuries of research knowledge. There are plethora of approaches that might be of great interest and of great applicability in the context of protein function annotation. For instance, in GrAPFI, we proposed an automatic function inference technique that works on a graph of proteins linked by their domain similarity. We used a relatively simple but effective neighborhood-based label propagation that considers only the local directly connected neighbors during transferring of the annotations. However, there is further scope of exploration based on subgraphs, for example, triangles and clicks, among the neighbors. Presence of sub-structures among the neighbors can significantly re-assure the presence of strong affinity and thus can improve the confidence score of the annotation.

The order of the domains has not been considered in GrAPFI. A possible extension can be

to explore ordered sets of domains to connect the proteins in the graph. GrAPFI is implemented in a way such that it can be used for large datasets without further complexity. In the current implementation, GrAPFI uses only the first order neighbors to transfer the annotations. Therefore, the computation of the annotation score does not require to build the complete graph which greatly reduces the computational complexity. Moreover, it computes the neighbors by maintaining two databases : 1) protein to domains : this contains proteins and their domains, 2) domains to proteins : this contains domains and proteins that contain this domain. However, exploring higher order neighbors requires to build the complete graph which is computationally expensive for large dataset. Furthermore, large scale annotation can be achieved using big data technologies such as using scalable graph processing platforms like SPARK, HADOOP etc. In Appendix 2, Figure 1 describes a pipeline based on SPARK implementation to process large scale graph for functional annotation of proteins. Similarly, GO annotation discussed in Chapter 4 requires further exploration of semantic similarity approaches to establish the efficacy of the pruning and hierarchical post-processing technique that is proposed in the chapter. There is also scope of using advanced representation learning for computing embeddings of the GO ontology terms. These embeddings eventually can be utilized to find semantic similarity and thus for finding membership scores of the GO terms.

Domain embedding introduced in Chapter 5 is an important direction towards the solution. The results section shows a promising outcome in contrast to the 3-mer embeddings. In this work, we have used one of the primitive word embedding techniques. Therefore, an explorative future ambition could be to integrate more advanced NLP-based model with a focus on GO annotation.

In our last contribution, we proposed Prot-A-GAN that models the problem of automatic protein function annotation as a link prediction task in a knowledge graph. We proposed a GAN inspired knowledge graph embedding approach. The rationale of using GAN-like training in a knowledge graph setting lies in the requirement of automatic protein function annotation task. In this task, given a protein, we want to know its function i.e. we are interested to generate annotations for proteins. However, we can not generate discrete samples. Therefore, it is more appropriate to say that we want to select appropriate annotations from a pool of predefined annotations by exploring the links in the knowledge graph. Therefore, we want to train a model that would do knowledge graph embedding as well as would select right annotation leveraging those embeddings. In the above mentioned scenario, generative machine learning models serve the purpose best. As GAN models are already explored in knowledge graph embedding modeling, we took this ambition to use GAN-like training improvised by using biomedical knowledge, customized negative sampling, and target-specific random walk. The prototype shows promising outcome in function annotation task. However, a detailed theoretical understanding as well as empirical treatment are necessary for further proving the efficacy of the approach. Moreover, Prot-A-GAN takes inspiration from generative models, reinforcement learning, and random walks. These are already rich areas with numerous success stories. A possible future direction could be to explore these variations, settings, and techniques. Hyper-parameter tuning is crucial for best outcome. Training GAN is already challenging. Random walk adds further computational complexity. Reducing the computational complexity for large-scale training could also be another future ambition to face the practical challenge of the task.

Automatic functional annotation of proteins is an open research problem that exists for decades now. Associating appropriate functions, for example, GO terms or EC numbers, to protein sequence, is a long-hold problem in biology. Decades of research have accumulated a number of approaches that explore various facets of proteins annotation problem. Some methods work directly on protein sequences, other uses structural and physico-chemical properties. Domains - the conserved regions in protein sequence - are important features linked to the protein functions.

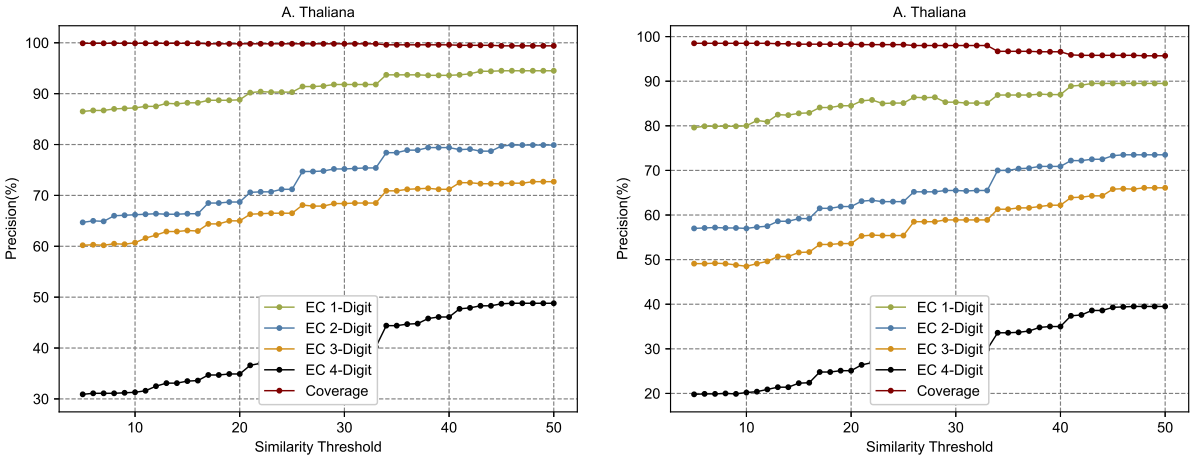
Currently deployed annotation pipeline in UniProtKB uses domains and taxonomic information for finding annotations. Realizing the importance of domains in protein function annotation, in this thesis, we proposed graph-based approaches that harness the significance of domains for automatic functional annotation of proteins. Through empirical experiments, we have shown the promises of the proposed methods. Despite the fact that GrAPFI and domain embeddings performed with high precision, recall and F-measure, there are still scope of improvements and further exploration. Prot-A-GAN opens up a new avenue for using adversarial and generative models for proteins-to-GO-term annotation and knowledge discovery from biomedical knowledge graph as well.

Cinquième partie

Appendix

Appendix 1

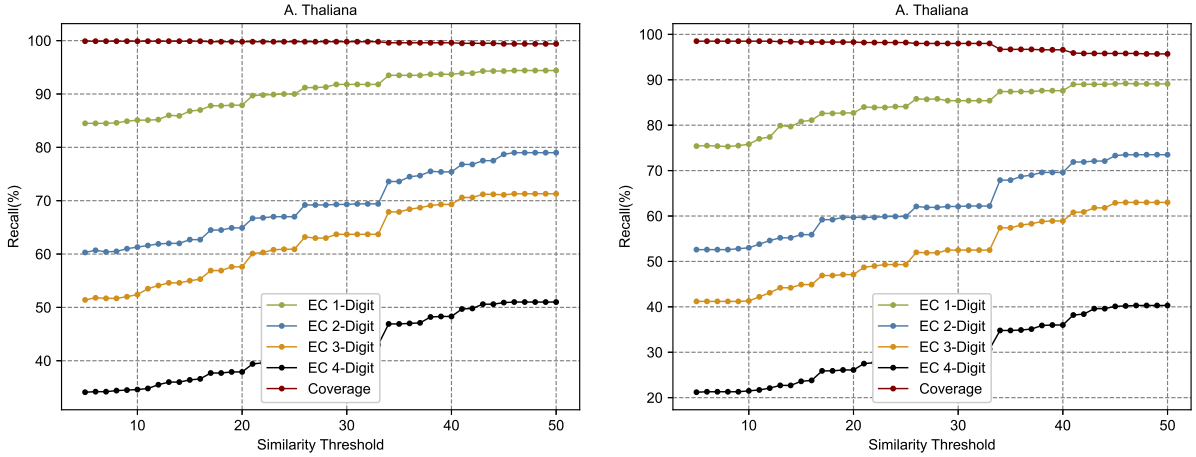
1 Experimental results from Grapfi



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

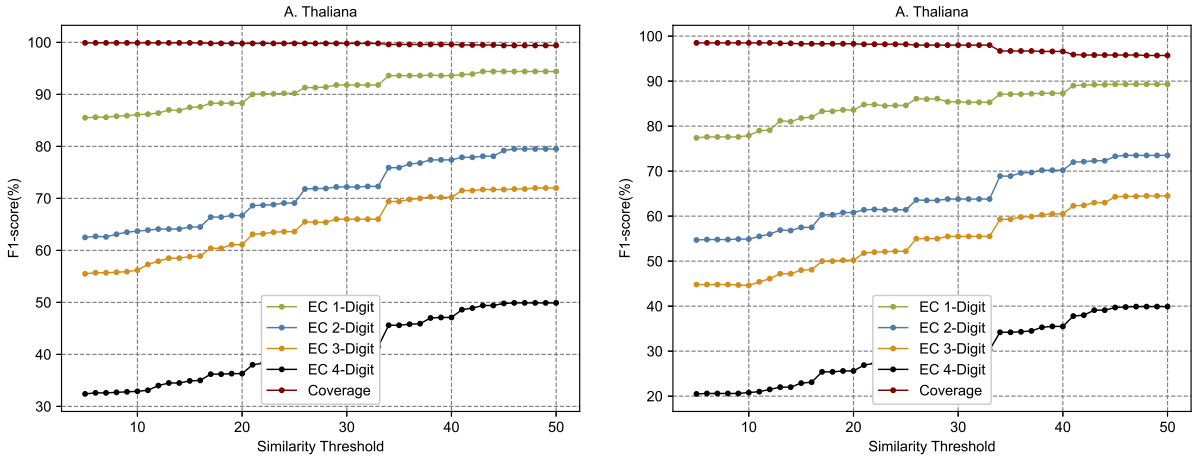
FIGURE 1 – The precision and coverage for different similarity thresholds for *A. Thaliana* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

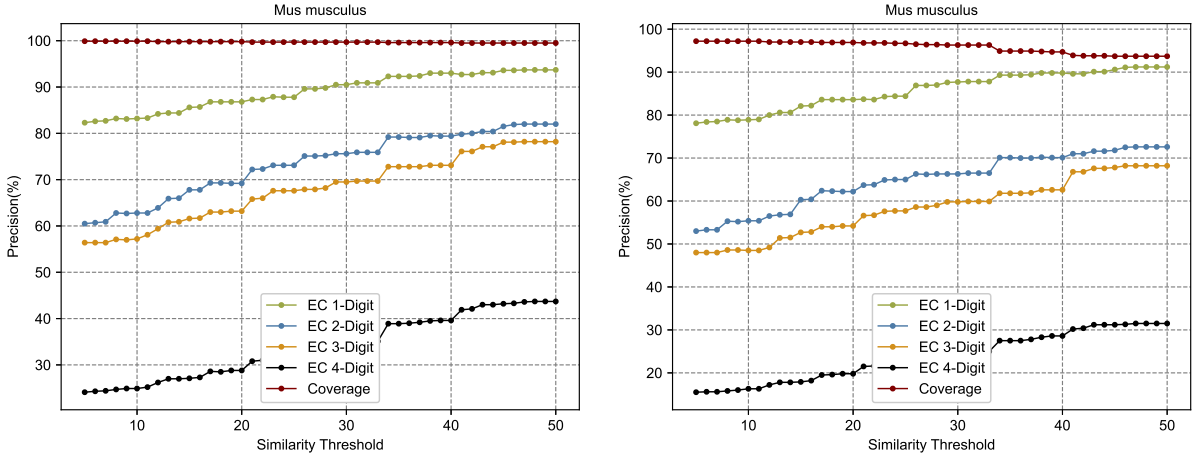
FIGURE 2 – The recall and coverage for different similarity thresholds for *A. Thaliana* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

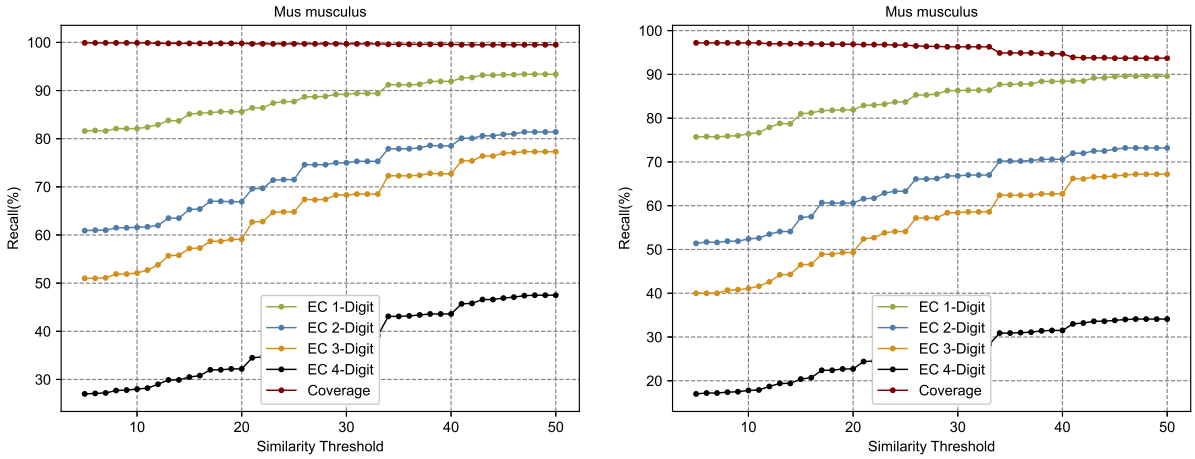
FIGURE 3 – The F1 score and coverage for different similarity thresholds for *A. Thaliana* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

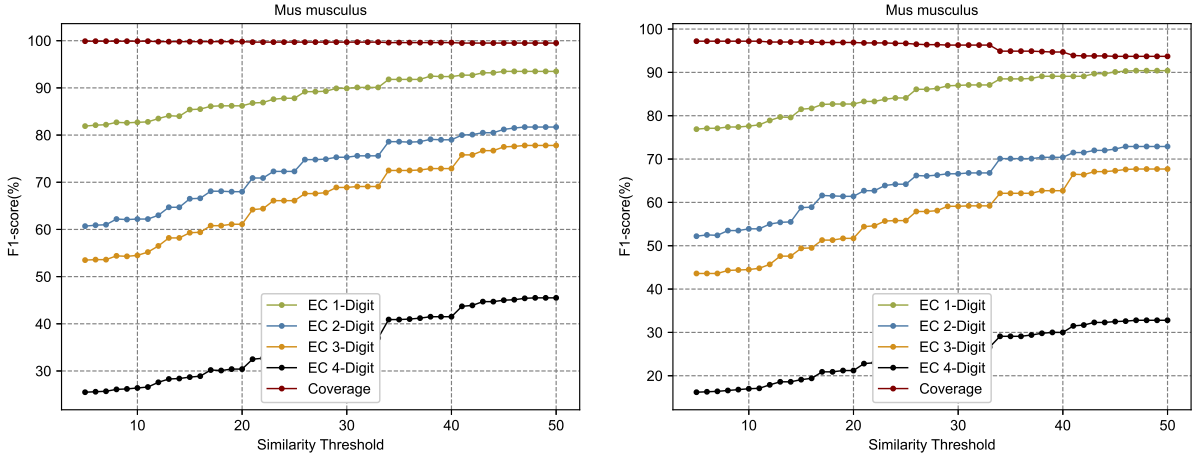
FIGURE 4 – The precision and coverage for different similarity thresholds for *Mus musculus* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

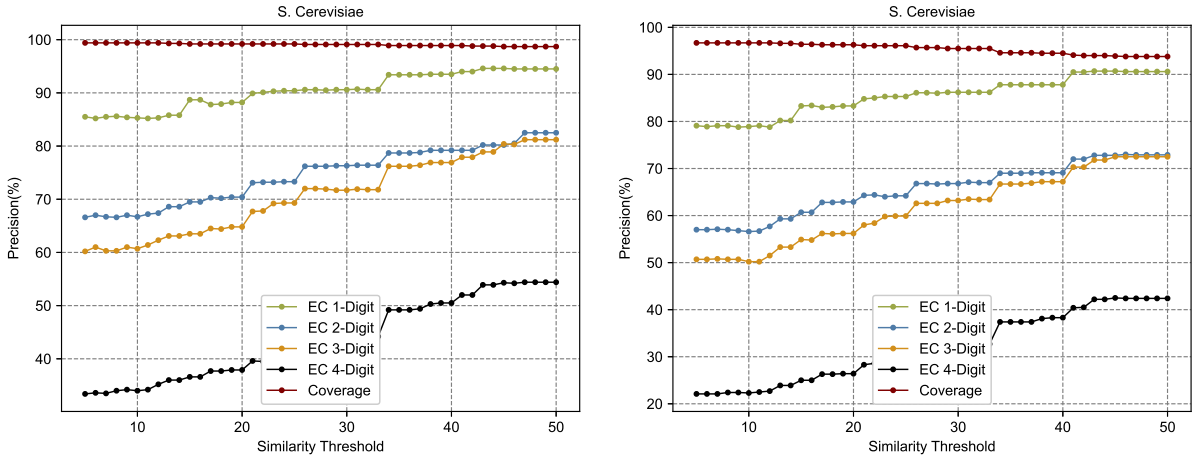
FIGURE 5 – The recall and coverage for different similarity thresholds for *Mus musculus* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

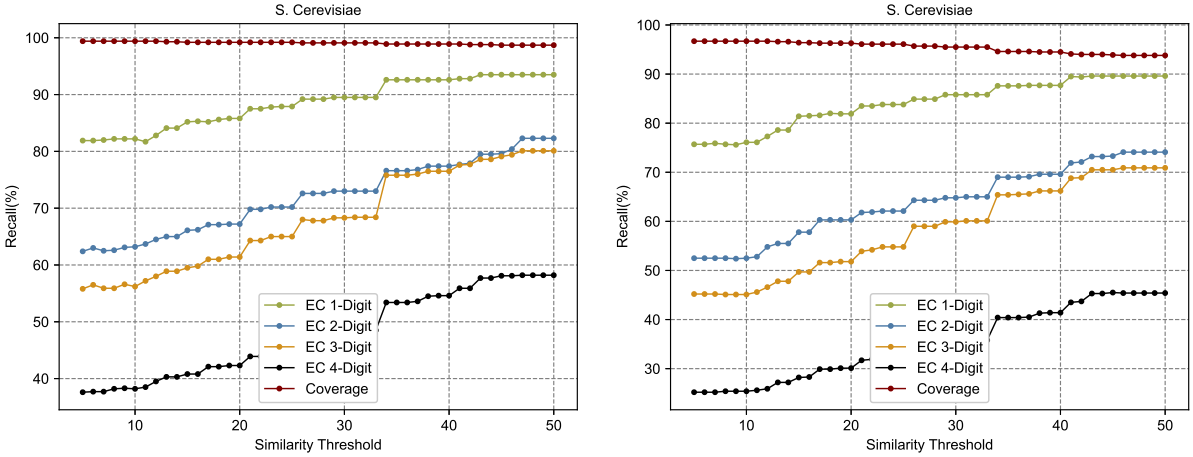
FIGURE 6 – The F1 score and coverage for different similarity thresholds for *Mus musculus* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

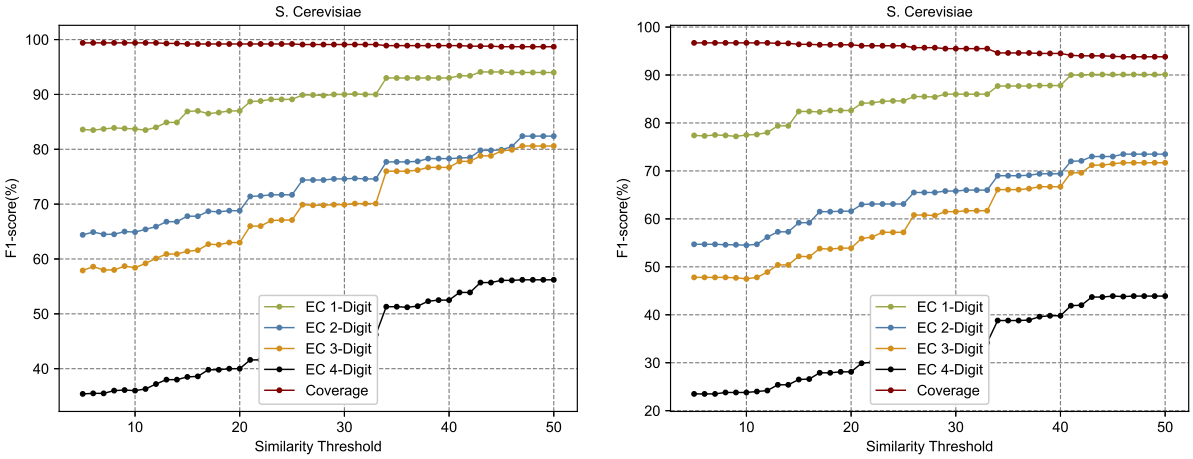
FIGURE 7 – The precision and coverage for different similarity thresholds for *Saccharomyces cerevisiae* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

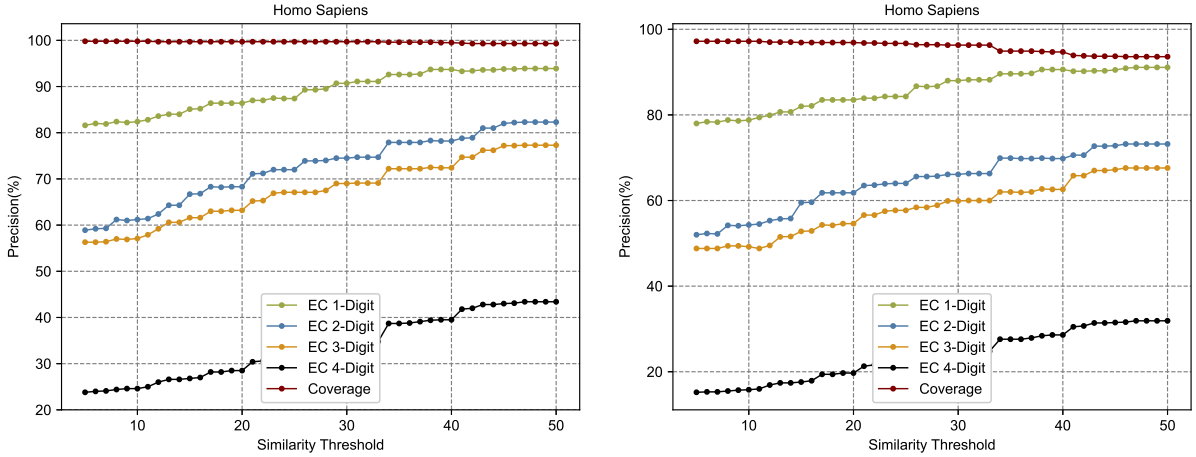
FIGURE 8 – The recall and coverage for different similarity thresholds for *Saccharomyces cerevisiae* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

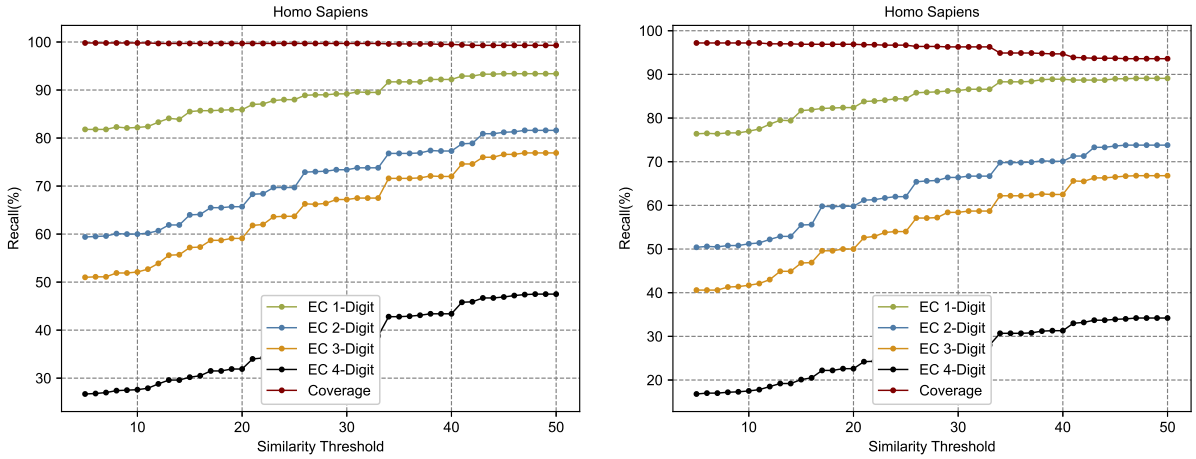
FIGURE 9 – The F1 score and coverage for different similarity thresholds for *Saccharomyces cerevisiae* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

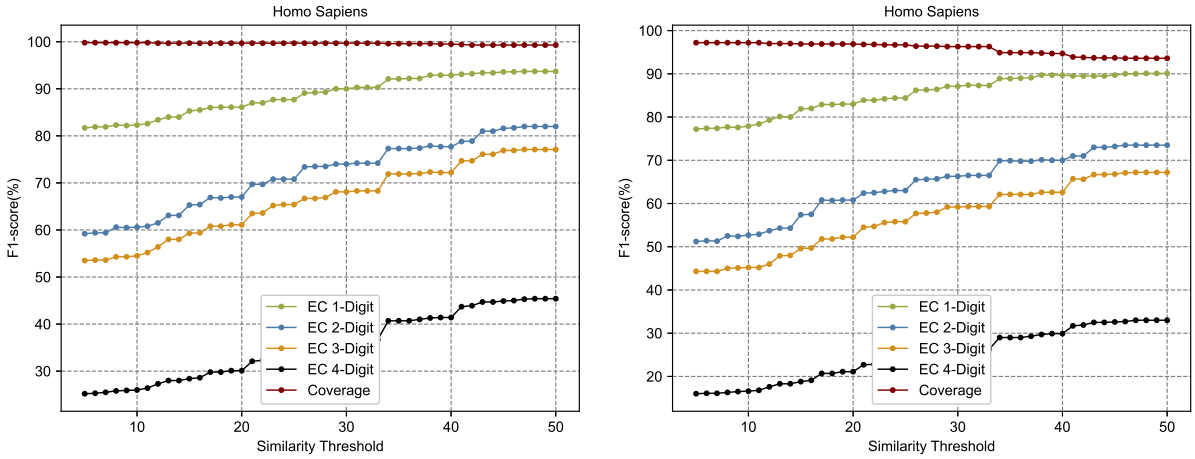
FIGURE 10 – The precision and coverage for different similarity thresholds for *Homo sapiens* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

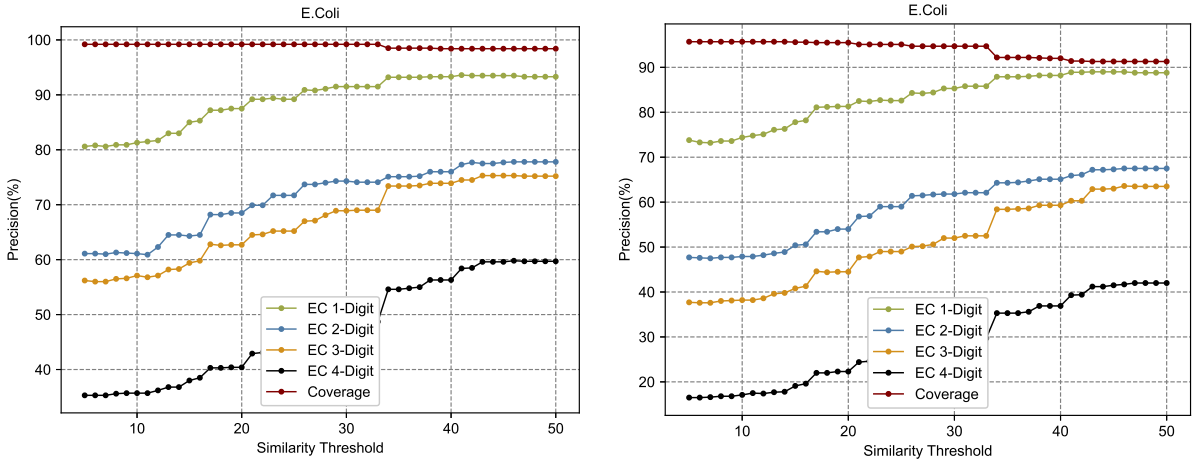
FIGURE 11 – The recall and coverage for different similarity thresholds for *Homo sapiens* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

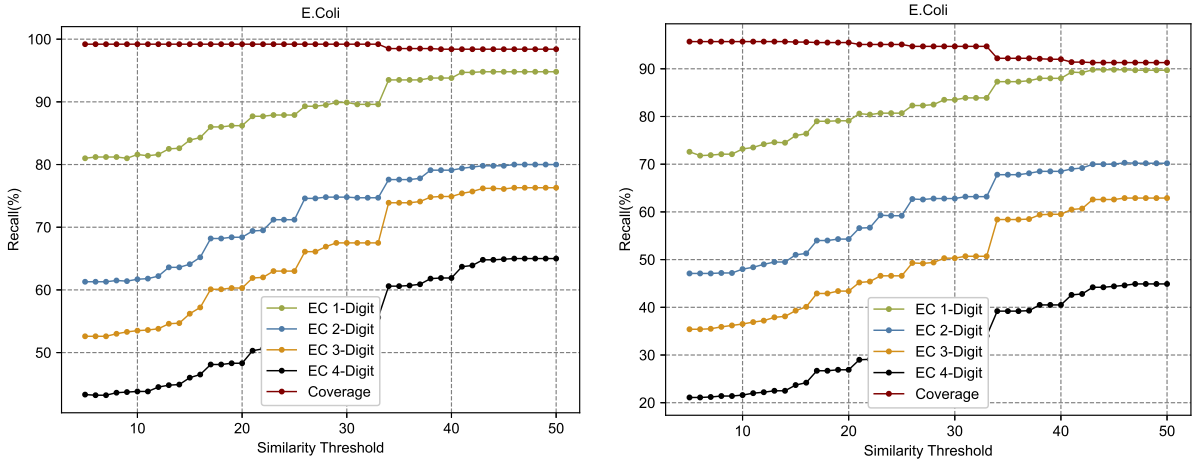
FIGURE 12 – The F1 score and coverage for different similarity thresholds for *Homo sapiens* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

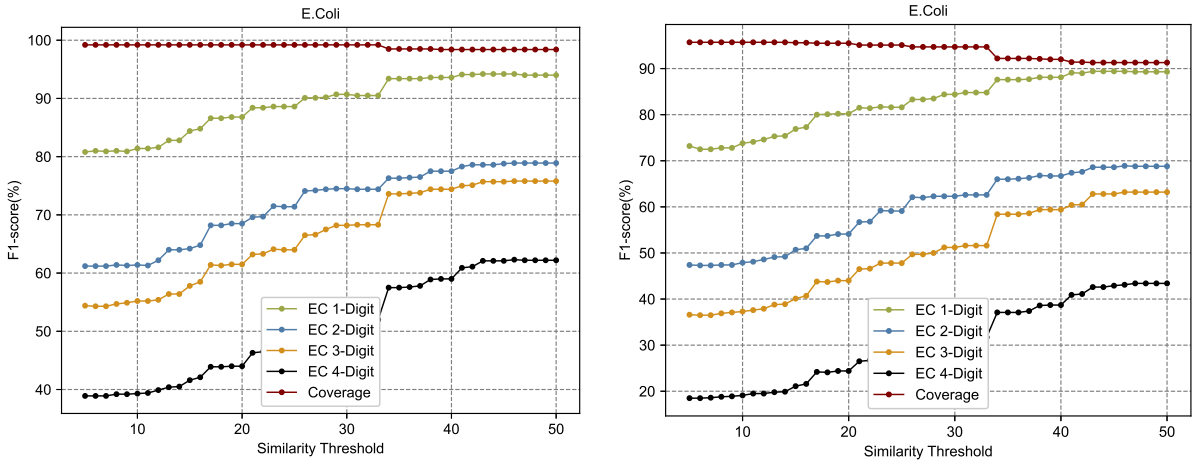
FIGURE 13 – The precision and coverage for different similarity thresholds for *E. Coli* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

FIGURE 14 – The recall and coverage for different similarity thresholds for *Homo sapiens* reference proteome



(a) Upper similarity threshold is less than or equal to 1 i.e. counting on exact match as well.

(b) Upper similarity threshold is less than 1 i.e. ignoring exact match in domain composition

FIGURE 15 – The F1 score and coverage for different similarity thresholds for *E. Coli* reference proteome

Appendix 2

1 Distributed Framework for GrAPFI

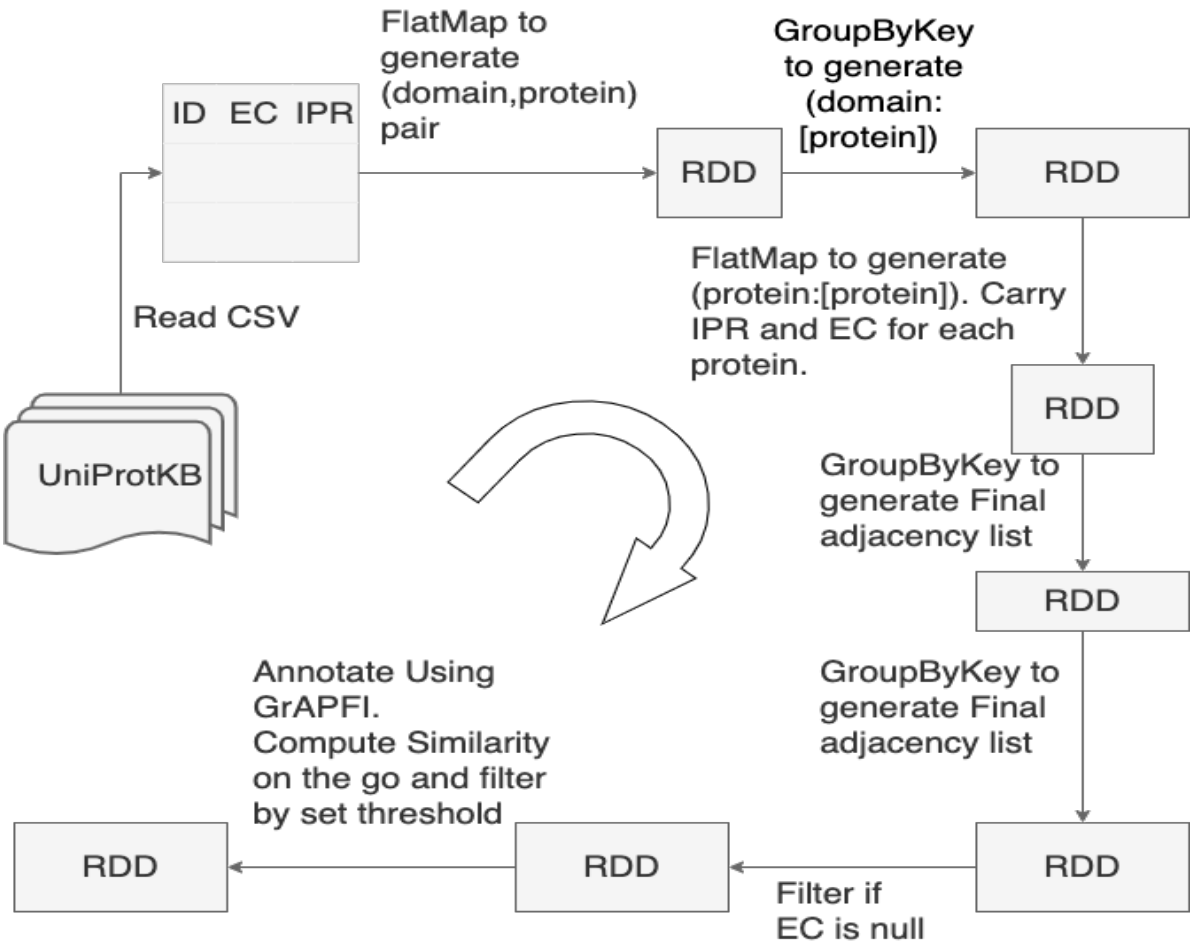


FIGURE 1 – Distributed Framework for GrAPFI.

Bibliographie

[Akiva et al., 2014] Akiva, E., Brown, S., Almonacid, D. E., Barber 2nd, A. E., Custer, A. F., Hicks, M. A., Huang, C. C., Lauck, F., Mashiyama, S. T., Meng, E. C., et al. (2014). The structure–function linkage database. *Nucleic acids research*, 42(D1) :D521–D530.

[Altschul et al., 1997] Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped blast and psi-blast : a new generation of protein database search programs. *Nucleic Acids Research*, 25(17) :3389–3402.

[Arakaki et al., 2009] Arakaki, A. K., Huang, Y., and Skolnick, J. (2009). Eficaz 2 : enzyme function inference by a combined approach enhanced by machine learning. *BMC bioinformatics*, 10(1) :107.

[Asgari and Mofrad, 2015] Asgari, E. and Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11) :e0141287.

[Ashburner and et al., 2000] Ashburner, M. and et al. (2000). Gene ontology : tool for the unification of biology. *Nature genetics*, 25(1) :25.

[Attwood et al., 2012] Attwood, T. K., Coletta, A., Muirhead, G., Pavlopoulou, A., Philippou, P. B., Popov, I., Roma-Mateo, C., Theodosiou, A., and Mitchell, A. L. (2012). The prints database : a fine-grained protein sequence annotation and analysis resource—its status in 2012. *Database*, 2012.

[Barabási, 2003] Barabási, A.-L. (2003). Linked : The new science of networks.

[Blum et al., 2021] Blum, M., Chang, H.-Y., Chuguransky, S., Grego, T., Kandasaamy, S., Mitchell, A., Nuka, G., Paysan-Lafosse, T., Qureshi, M., Raj, S., et al. (2021). The interpro protein families and domains database : 20 years on. *Nucleic Acids Research*, 49(D1) :D344–D354.

[Bordes et al., 2014] Bordes, A., Glorot, X., Weston, J., and Bengio, Y. (2014). A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2) :233–259.

[Bordes et al., 2013] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

[Boudellioua et al., 2016] Boudellioua, I., Saidi, R., Hoehndorf, R., Martin, M. J., and Solovyev, V. (2016). Prediction of metabolic pathway involvement in prokaryotic uniprotkb data by association rule mining. *PloS one*, 11(7) :e0158896.

[Cai et al., 2004] Cai, C., Han, L., Ji, Z., and Chen, Y. (2004). Enzyme family classification by support vector machines. *Proteins : Structure, Function, and Bioinformatics*, 55(1) :66–76.

[Cai et al., 2003] Cai, C., Han, L., Ji, Z. L., Chen, X., and Chen, Y. Z. (2003). Svm-prot : web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic acids research*, 31(13) :3692–3697.

[Cai and Wang, 2018] Cai, L. and Wang, W. Y. (2018). Kbgan : Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, pages 1470–1480.

[Cai and Chou, 2005] Cai, Y.-D. and Chou, K.-C. (2005). Predicting enzyme subclass by functional domain composition and pseudo amino acid composition. *Journal of Proteome Research*, 4(3) :967–971.

[Che et al., 2017] Che, Z., Purushotham, S., Khemani, R., and Liu, Y. (2017). Interpretable deep models for ICU outcome prediction. *AMIA Annual Symposium proceedings*, 2016 :371–380.

[Choi et al., 2017] Choi, E., Bahadori, M. T., Song, L., Stewart, W. F., and Sun, J. (2017). Gram : graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 787–795.

[Chou, 2009] Chou, K.-C. (2009). Pseudo amino acid composition and its applications in bioinformatics, proteomics and system biology. *Current Proteomics*, 6(4) :262–274.

[Chua et al., 2006] Chua, H. N., Sung, W.-K., and Wong, L. (2006). Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. *Bioinformatics*, 22(13) :1623–1630.

[Conesa et al., 2005] Conesa, A., Götz, S., García-Gómez, J. M., Terol, J., Talón, M., and Robles, M. (2005). Blast2go : a universal tool for annotation, visualization and analysis in functional genomics research. *Bioinformatics*, 21(18) :3674–3676.

[Cornish-Bowden, 2014] Cornish-Bowden, A. (2014). Current IUBMB recommendations on enzyme nomenclature and kinetics. *Perspectives in Science*, 1(1-6) :74–87.

[Dalkiran et al., 2018] Dalkiran, A., Rifaioglu, A. S., Martin, M. J., Cetin-Atalay, R., Atalay, V., and Doğan, T. (2018). ECPred : a tool for the prediction of the enzymatic functions of protein sequences based on the EC nomenclature. *BMC Bioinformatics*, 19(1) :334.

[des Jardins et al., 1997] des Jardins, M., Karp, P. D., Krummenacker, M., Lee, T. J., and Ouzounis, C. A. (1997). Prediction of enzyme classification from protein sequence without the use of sequence similarity. In *Proc Int Conf Intell Syst Mol Biol*, volume 5, pages 92–99.

[Ding et al., 2018] Ding, M., Tang, J., and Zhang, J. (2018). Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 913–922.

[Dobson and Doig, 2005] Dobson, P. D. and Doig, A. J. (2005). Predicting enzyme class from protein structure without alignments. *Journal of molecular biology*, 345(1) :187–199.

[Dumontier et al., 2014] Dumontier, M., Callahan, A., Cruz-Toledo, J., Ansell, P., Emonet, V., Belleau, F., and Droit, A. (2014). Bio2rdf release 3 : a larger connected network of linked data for the life sciences. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track*, volume 1272, pages 401–404. Citeseer.

[El-Gebali et al., 2019] El-Gebali, S., Mistry, J., Bateman, A., Eddy, S. R., Luciani, A., Potter, S. C., Qureshi, M., Richardson, L. J., Salazar, G. A., Smart, A., et al. (2019). The pfam protein families database in 2019. *Nucleic acids research*, 47(D1) :D427–D432.

[Ferrucci et al., 2010] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., et al. (2010). Building watson : An overview of the deepqa project. *AI magazine*, 31(3) :59–79.

[Finn et al., 2011] Finn, R. D., Clements, J., and Eddy, S. R. (2011). HMMER web server : interactive sequence similarity searching. *Nucleic Acids Research*, 39(2) :W29–W37.

[Fu et al., 2012] Fu, L., Niu, B., Zhu, Z., Wu, S., and Li, W. (2012). Cd-hit : accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23) :3150–3152.

[Gattiker et al., 2003] Gattiker, A., Michoud, K., Rivoire, C., Auchincloss, A. H., Coudert, E., Lima, T., Kersey, P., Pagni, M., Sigrist, C. J., Lachaize, C., Veuthey, A.-L., Gasteiger, E., and Bairoch, A. (2003). Automated annotation of microbial proteomes in SWISS-PROT. *Computational Biology and Chemistry*, 27(1) :49–58.

[Gong et al., 2016] Gong, Q., Ning, W., and Tian, W. (2016). Gofdr : a sequence alignment based method for predicting protein functions. *Methods*, 93 :3–14.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[Grover and Leskovec, 2016] Grover, A. and Leskovec, J. (2016). node2vec : Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

[Gysi et al., 2020] Gysi, D. M., Valle, Í. D., Zitnik, M., Ameli, A., Gan, X., Varol, O., Sanchez, H., Baron, R. M., Ghiassian, D., Loscalzo, J., et al. (2020). Network medicine framework for identifying drug repurposing opportunities for covid-19. *arXiv preprint arXiv :2004.07229*.

[Haft et al., 2012] Haft, D. H., Selengut, J. D., Richter, R. A., Harkins, D., Basu, M. K., and Beck, E. (2012). Tigrfams and genome properties in 2013. *Nucleic acids research*, 41(D1) :D387–D395.

[Himmelstein et al., 2017] Himmelstein, D. S., Lizee, A., Hessler, C., Brueggeman, L., Chen, S. L., Hadley, D., Green, A., Khankhanian, P., and Baranzini, S. E. (2017). Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *Elife*, 6 :e26726.

[Hishigaki et al., 2001] Hishigaki, H., Nakai, K., Ono, T., Tanigami, A., and Takagi, T. (2001). Assessment of prediction accuracy of protein function from protein–protein interaction data. *Yeast*, 18(6) :523–531.

[Hu et al., 2019] Hu, K., Liu, H., and Hao, T. (2019). A knowledge selective adversarial network for link prediction in knowledge graph. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 171–183. Springer.

[Huang et al., 2007] Huang, W.-L., Chen, H.-M., Hwang, S.-F., and Ho, S.-Y. (2007). Accurate prediction of enzyme subfamily class using an adaptive fuzzy k-nearest neighbor method. *Biosystems*, 90(2) :405–413.

[Jiang and et al., 2016] Jiang, Y. and et al. (2016). An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome biology*, 17(1) :184.

[Jones et al., 2014] Jones, P., Binns, D., Chang, H.-Y., Fraser, M., Li, W., McAnulla, C., McWilliam, H., Maslen, J., Mitchell, A., Nuka, G., et al. (2014). Interproscan 5 : genome-scale protein function classification. *Bioinformatics*, 30(9) :1236–1240.

[Jones and et al., 2014] Jones, P. and et al. (2014). Interproscan 5 : genome-scale protein function classification. *Bioinformatics*, 30(9) :1236–1240.

[Joulin et al., 2016] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016). Fasttext.zip : Compressing text classification models. *arXiv preprint arXiv :1612.03651*.

[Joulin et al., 2017] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

[Kimothi et al., 2016] Kimothi, D., Soni, A., Biyani, P., and Hogan, J. M. (2016). Distributed representations for biological sequence analysis. *arXiv preprint arXiv :1608.05949*.

[Koskinen et al., 2015] Koskinen, P., Törönen, P., Nokso-Koivisto, J., and Holm, L. (2015). Pannzer : high-throughput functional annotation of uncharacterized proteins in an error-prone environment. *Bioinformatics*, 31(10) :1544–1552.

[Kretschmann et al., 2001] Kretschmann, E., Fleischmann, W., and Apweiler, R. (2001). Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on SWISS-PROT. *Bioinformatics*, 17 10 :920–6.

[Krompaß et al., 2015] Krompaß, D., Baier, S., and Tresp, V. (2015). Type-constrained representation learning in knowledge graphs. In *International semantic web conference*, pages 640–655. Springer.

[Kulmanov and Hoehndorf, 2020] Kulmanov, M. and Hoehndorf, R. (2020). Deepgoplus : improved protein function prediction from sequence. *Bioinformatics*, 36(2) :422–429.

[Kulmanov et al., 2017] Kulmanov, M., Khan, M. A., and Hoehndorf, R. (2017). Deepgo : predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4) :660–668.

[Kumar and Skolnick, 2012] Kumar, N. and Skolnick, J. (2012). Eficaz2. 5 : application of a high-precision enzyme function predictor to 396 proteomes. *Bioinformatics*, 28(20) :2687–2688.

[Kummerfeld and Teichmann, 2009] Kummerfeld, S. K. and Teichmann, S. A. (2009). Protein domain organisation : adding order. *BMC Bioinformatics*, 10(1) :39.

[Letunic and Bork, 2018] Letunic, I. and Bork, P. (2018). 20 years of the smart protein domain annotation resource. *Nucleic acids research*, 46(D1) :D493–D496.

[Li et al., 2018] Li, Y., Wang, S., Umarov, R., Xie, B., Fan, M., Li, L., and Gao, X. (2018). DEEPre : sequence-based enzyme EC number prediction by deep learning. *Bioinformatics*, 34(5) :760–769.

[Li et al., 2016] Li, Y. H., Xu, J. Y., Tao, L., Li, X. F., Li, S., Zeng, X., Chen, S. Y., Zhang, P., Qin, C., Zhang, C., et al. (2016). Svm-prot 2016 : a web-server for machine learning prediction of protein functional families from sequence irrespective of similarity. *PloS one*, 11(8) :e0155290.

[Lim et al., 2018] Lim, T., Schaffner, T., Crovella, M., et al. (2018). A multi-species functional embedding integrating sequence and network structure. In *Research in Computational Molecular Biology–22nd Annual International Conference, RECOMB*, pages 263–265. Springer.

[Lu et al., 2007] Lu, L., Qian, Z., Cai, Y.-D., and Li, Y. (2007). ECS : an automatic enzyme classifier based on functional domain composition. *Computational Biology and Chemistry*, 31(3) :226–232.

[Lu et al., 2020] Lu, S., Wang, J., Chitsaz, F., Derbyshire, M. K., Geer, R. C., Gonzales, N. R., Gwadz, M., Hurwitz, D. I., Marchler, G. H., Song, J. S., et al. (2020). Cdd/sparcle : the conserved domain database in 2020. *Nucleic acids research*, 48(D1) :D265–D268.

[Matsuda et al., 2005] Matsuda, S., Vert, J.-P., Saigo, H., Ueda, N., Toh, H., and Akutsu, T. (2005). A novel representation of protein sequences for prediction of subcellular location using support vector machines. *Protein Science*, 14(11) :2804–2813.

[Medlar et al., 2018] Medlar, A. J., Törönen, P., Zosa, E., and Holm, L. (2018). Pannzer 2 : Annotate a complete proteome in minutes ! *Nucl. Acids Res*, 43 :W24–W29.

[Melidis and Nejdl, 2021] Melidis, D. P. and Nejdl, W. (2021). Capturing protein domain structure and function using self-supervision on domain architectures. *Algorithms*, 14(1) :28.

[Mi et al., 2019] Mi, H., Muruganujan, A., Ebert, D., Huang, X., and Thomas, P. D. (2019). Panther version 14 : more genomes, a new panther go-slim and improvements in enrichment analysis tools. *Nucleic acids research*, 47(D1) :D419–D426.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Miller, 1995] Miller, G. A. (1995). Wordnet : a lexical database for english. *Communications of the ACM*, 38(11) :39–41.

[Mitchell et al., 2018a] Mitchell, A. L., Attwood, T. K., Babbitt, P. C., Blum, M., Bork, P., Bridge, A., Brown, S. D., Chang, H.-Y., El-Gebali, S., Fraser, M. I., et al. (2018a). Interpro in 2019 : improving coverage, classification and access to protein sequence annotations. *Nucleic acids research*, 47(D1) :D351–D360.

[Mitchell et al., 2018b] Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., et al. (2018b). Never-ending learning. *Communications of the ACM*, 61(5) :103–115.

[Mohamed et al., 2020] Mohamed, S. K., Nounu, A., and Nováček, V. (2020). Biological applications of knowledge graph embedding models. *Briefings in Bioinformatics*.

[Mohamed et al., 2019] Mohamed, S. K., Novácek, V., Vandenbussche, P.-Y., and Muñoz, E. (2019). Loss functions in knowledge graph embedding models. In *DL4KG@ ESWC*, pages 1–10.

[Mount, 2007] Mount, D. W. (2007). Using the basic local alignment search tool (blast). *Cold Spring Harbor Protocols*, 2007(7) :pdb–top17.

[Nabieva et al., 2005] Nabieva, E., Jim, K., Agarwal, A., Chazelle, B., and Singh, M. (2005). Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *Bioinformatics*, 21(suppl_1) :i302–i310.

[Nagao Chioko and Kenji, 2014] Nagao Chioko, N. N. and Kenji, M. (2014). Prediction of detailed enzyme functions and identification of specificity determining residues by random forests. *PLoS One*, 9(1).

[Nasibov and Kandemir-Cavas, 2009] Nasibov, E. and Kandemir-Cavas, C. (2009). Efficiency analysis of KNN and minimum distance-based classifiers in enzyme family prediction. *Computational Biology and Chemistry*, 33(6) :461–464.

[Nelson et al., 2019] Nelson, W., Zitnik, M., Wang, B., Leskovec, J., Goldenberg, A., and Sharan, R. (2019). To embed or not : network embedding as a paradigm in computational biology. *Frontiers in genetics*, 10 :381.

[Nicholson and Greene, 2020] Nicholson, D. N. and Greene, C. S. (2020). Constructing knowledge graphs and their biomedical applications. *Computational and structural biotechnology journal*, 18 :1414.

[Nickel et al., 2011] Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 809–816.

[Nickel et al., 2012] Nickel, M., Tresp, V., and Kriegel, H.-P. (2012). Factorizing yago : scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280.

[Nikolskaya et al., 2006] Nikolskaya, A. N., Arighi, C. N., Huang, H., Barker, W. C., and Wu, C. H. (2006). Pirsf family classification system for protein functional and evolutionary analysis. *Evolutionary Bioinformatics*, 2 :117693430600200033.

[Pandurangan et al., 2019] Pandurangan, A. P., Stahlhacke, J., Oates, M. E., Smithers, B., and Gough, J. (2019). The superfamily 2.0 database : a significant proteome update and a new webserver. *Nucleic acids research*, 47(D1) :D490–D494.

[Pedruzzi et al., 2015] Pedruzzi, I., Rivoire, C., Auchincloss, A. H., Coudert, E., Keller, G., De Castro, E., Baratin, D., Cuche, B. A., Bougueleret, L., Poux, S., et al. (2015). Hamap in 2015 : updates to the protein family classification and annotation system. *Nucleic acids research*, 43(D1) :D1064–D1070.

[Perozzi et al., 2014] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk : Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

[Qian, 2013] Qian, R. (2013). Understand your world with bing.(2013). *Bing Blogs*.

[Quester and Schomburg, 2011] Quester, S. and Schomburg, D. (2011). EnzymeDetector : an integrated enzyme function prediction tool and database. *BMC Bioinformatics*, 12(1) :376.

[Quevillon and et al., 2005] Quevillon, E. and et al. (2005). Interproscan : protein domains identifier. *Nucleic acids research*, 33(suppl_2) :W116–W120.

[Quevillon et al., 2005] Quevillon, E., Silventoinen, V., Pillai, S., Harte, N., Mulder, N., Apweiler, R., and Lopez, R. (2005). InterProScan : protein domains identifier. *Nucleic Acids Research*, 33(suppl_2) :W116–W120.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Mach. Learn.*, 1(1) :81–106.

[Radivojac and et al., 2013] Radivojac, P. and et al. (2013). A large-scale evaluation of computational protein function prediction. *Nature methods*, 10(3) :221.

[Rahman et al., 2014] Rahman, S. A., Cuesta, S. M., Furnham, N., Holliday, G. L., and Thornton, J. M. (2014). EC-BLAST : a tool to automatically search and compare enzyme reactions. *Nature Methods*, 11(2) :171.

[Ristoski and Paulheim, 2016] Ristoski, P. and Paulheim, H. (2016). Rdf2vec : Rdf graph embeddings for data mining. In *International Semantic Web Conference*, pages 498–514. Springer.

[Rossi et al., 2020] Rossi, A., Firmani, D., Matinata, A., Merialdo, P., and Barbosa, D. (2020). Knowledge graph embedding for link prediction : A comparative analysis. *arXiv preprint arXiv :2002.00819*.

[Roy et al., 2012] Roy, A., Yang, J., and Zhang, Y. (2012). Cofactor : an accurate comparative algorithm for structure-based protein function annotation. *Nucleic acids research*, 40(W1) :W471–W477.

[Saidi et al., 2017] Saidi, R., Boudellioua, I., Martin, M. J., and Solovyev, V. (2017). Rule mining techniques to predict prokaryotic metabolic pathways. In *Biological Networks and Pathway Analysis*, pages 311–331. Springer.

[Schwikowski et al., 2000] Schwikowski, B., Uetz, P., and Fields, S. (2000). A network of protein–protein interactions in yeast. *Nature biotechnology*, 18(12) :1257.

[Shen and Chou, 2007] Shen, H.-B. and Chou, K.-C. (2007). Ezypred : a top–down approach for predicting enzyme functional classes and subclasses. *Biochemical and biophysical research communications*, 364(1) :53–59.

[Shen et al., 2017] Shen, Z., Zhang, Y.-H., Han, K., Nandi, A. K., Honig, B., and Huang, D.-S. (2017). mirna-disease association prediction with collaborative matrix factorization. *Complexity*, 2017.

[Sigrist et al., 2012] Sigrist, C. J., De Castro, E., Cerutti, L., Cuche, B. A., Hulo, N., Bridge, A., Bougueleret, L., and Xenarios, I. (2012). New and continuing developments at prosite. *Nucleic acids research*, 41(D1) :D344–D347.

[Sillitoe et al., 2019] Sillitoe, I., Dawson, N., Lewis, T. E., Das, S., Lees, J. G., Ashford, P., Tolulope, A., Scholes, H. M., Senatorov, I., Bujan, A., et al. (2019). Cath : expanding the horizons of structure-based functional annotations for genome sequences. *Nucleic acids research*, 47(D1) :D280–D284.

[Sun and Han, 2013] Sun, Y. and Han, J. (2013). Mining heterogeneous information networks : a structural analysis approach. *Acm Sigkdd Explorations Newsletter*, 14(2) :20–28.

[The UniProt Consortium, 2015] The UniProt Consortium (2015). UniProt : a hub for protein information. *Nucleic Acids Research*, 43(D204-D212).

[Tian et al., 2004] Tian, W., Arakaki, A. K., and Skolnick, J. (2004). Eficaz : a comprehensive approach for accurate genome-scale enzyme function inference. *Nucleic acids research*, 32(21) :6226–6239.

[Volpato et al., 2013] Volpato, V., Adelfio, A., and Pollastri, G. (2013). Accurate prediction of protein enzymatic class by n-to-1 neural networks. *BMC Bioinformatics*, 14(1) :S11.

[Wang et al., 2019] Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Li, W., Xie, X., and Guo, M. (2019). Learning graph representation with generative adversarial nets. *IEEE Transactions on Knowledge and Data Engineering*.

[Wang et al., 2017a] Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., and Zhang, D. (2017a). Irgan : A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 515–524.

[Wang et al., 2017b] Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017b). Knowledge graph embedding : A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12) :2724–2743.

[Yang et al., 2015] Yang, J., Yan, R., Roy, A., Xu, D., Poisson, J., and Zhang, Y. (2015). The i-tasser suite : protein structure and function prediction. *Nature methods*, 12(1) :7.

[Yu et al., 2009] Yu, C., Zavaljevski, N., Desai, V., and Reifman, J. (2009). Genome-wide enzyme annotation with precision control : Catalytic families (CatFam) databases. *Proteins : Structure, Function, and Bioinformatics*, 74(2) :449–460.

[Zhang et al., 2017] Zhang, C., Freddolino, P. L., and Zhang, Y. (2017). Cofactor : improved protein function prediction by combining structure, sequence and protein–protein interaction information. *Nucleic Acids Research*, 45(W1) :W291–W299.

[Zhang et al., 2018] Zhang, C., Zheng, W., Freddolino, P. L., and Zhang, Y. (2018). Metago : Predicting gene ontology of non-homologous proteins through low-resolution protein structure prediction and protein–protein network mapping. *Journal of molecular biology*, 430(15) :2256–2265.

[Zhao et al., 2016] Zhao, B., Hu, S., Li, X., Zhang, F., Tian, Q., and Ni, W. (2016). An efficient method for protein function annotation based on multilayer protein networks. *Human Genomics*, 10(1) :33.

[Zhao and Wang, 2018] Zhao, C. and Wang, Z. (2018). Gogo : An improved algorithm to measure the semantic similarity between gene ontology terms. *Scientific reports*, 8(1) :15107.

[Zitnik and Leskovec, 2017] Zitnik, M. and Leskovec, J. (2017). Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14) :i190–i198.

Résumé

Les progrès des technologies de séquençage génomique ont conduit à une croissance exponentielle du nombre de séquences protéiques dans les bases de données publiques. Il est important d'exploiter cette énorme quantité de données pour décrire les êtres vivants au niveau moléculaire, et ainsi mieux comprendre les processus pathologiques humains et accélérer la découverte de médicaments. Une condition préalable, cependant, est que toutes ces protéines soient annotées avec des propriétés fonctionnelles telles que les numéros de commission enzymatique (EC) ou les termes de l'ontologie « Gene Ontology » (GO). Aujourd'hui, seule une petite fraction des protéines est annotée fonctionnellement et examinée manuellement par des experts car c'est une tâche coûteuse, lente et chronophage. Le développement d'outils d'annotation automatique des protéines est la voie à suivre pour réduire l'écart entre séquences protéiques annotées et non annotées et produire des annotations fiables. Aucun outil déjà développés n'est pleinement satisfaisant. Seuls quelques-uns utilisent les approches à base de graphes et tiennent compte de la composition en domaines des protéines qui sont des régions conservées à travers les séquences protéiques de la même famille. Dans cette thèse, nous concevons et évaluons des approches à base de graphes pour effectuer l'annotation automatique des fonctions protéiques et nous explorons l'impact de l'architecture en domaines sur les fonctions protéiques. La première partie est consacrée à l'annotation de la fonction des protéines à l'aide d'un graphe de similarité de domaines et de techniques de propagation d'étiquettes (ou de labels) améliorées. Tout d'abord, nous présentons GrAPFI ("Graph-based Automatic Protein Function Inference") pour l'annotation automatique des protéines par les numéros EC et par des termes GO. Nous validons les performances de GrAPFI en utilisant six protéomes de référence dans UniprotKB/SwissProt, et nous comparons les résultats de GrAPFI avec des outils de référence. Nous avons constaté que GrAPFI atteint une meilleure précision et une couverture comparable ou meilleure par rapport aux outils existants. La deuxième partie traite de l'apprentissage de représentations pour les entités biologiques. Au début, nous nous concentrons sur les techniques de plongement lexical ("word embedding"), utilisant les réseaux neuronaux. Nous formulons la tâche d'annotation comme une tâche de classification de textes. Nous construisons un corpus de protéines sous forme de phrases composées de leurs domaines respectifs et nous apprenons une représentation vectorielle à dimension fixe. Ensuite, nous portons notre attention sur l'apprentissage de représentations à partir de graphes de connaissances intégrant différentes sources de données liées aux protéines et à leurs fonctions. Nous formulons le problème d'annotation fonctionnelle des protéines comme une tâche de prédiction de liens entre une protéine et un terme GO. Nous proposons Prot-A-GAN, un modèle d'apprentissage automatique inspiré des réseaux antagonistes génératifs (GAN pour "Generative Adversarial Network"). Nous observons que Prot-A-GAN fonctionne avec des résultats prometteurs pour associer des fonctions appropriées aux protéines requêtes. En conclusion, cette thèse revisite le problème crucial de l'annotation automatique des fonctions protéiques à grande échelle en utilisant des techniques innovantes d'intelligence artificielle. Elle ouvre de larges perspectives, notamment pour l'utilisation des graphes de connaissances, disponibles aujourd'hui dans de nombreux domaines autres que l'annotation de protéines grâce aux progrès de la science des données.

Mots-clés: science des données, Artificial Intelligence,Machine Learning, Big Data, Network Science, Interaction Network, Bioinformatics, Computational Biology, Knowledge Graph, Pro-

tein Annotation, Gene Ontology, GO prediction, Enzyme Commission Number, EC prediction, Automatic Function Annotation, Label Propagation, GrAPFI, Domain Similarity Graph, Representation Learning, Neural Network, Domain Embedding, Sequence Embedding, Heterogeneous Biological Network, Biomedical Knowledge Graph, Knowledge Graph Embedding, Prot-A-GAN, GAN, Generative Adversarial Network for Graph, Protein Annotation GAN

Abstract

Due to the recent advancement in genomic sequencing technologies, the number of protein entries in public databases is growing exponentially. It is important to harness this huge amount of data to describe living things at the molecular level, which is essential for understanding human disease processes and accelerating drug discovery. A prerequisite, however, is that all of these proteins be annotated with functional properties such as Enzyme Commission (EC) numbers and Gene Ontology (GO) terms. Today, only a small fraction of the proteins is functionally annotated and reviewed by expert curators because it is expensive, slow and time-consuming. Developing automatic protein function annotation tools is the way forward to reduce the gap between the annotated and unannotated proteins and to predict reliable annotations for unknown proteins. Many tools of this type already exist, but none of them are fully satisfactory. We observed that only few consider graph-based approaches and the domain composition of proteins. Indeed, domains are conserved regions across protein sequences of the same family. In this thesis, we design and evaluate graph-based approaches to perform automatic protein function annotation and we explore the impact of domain architecture on protein functions. The first part is dedicated to protein function annotation using domain similarity graph and neighborhood-based label propagation technique. We present GrAPFI (Graph-based Automatic Protein Function Inference) for automatically annotating proteins with enzymatic functions (EC numbers) and GO terms from a protein-domain similarity graph. We validate the performance of GrAPFI using six reference proteomes from UniprotKB/SwissProt and compare GrAPFI results with state-of-the-art EC prediction approaches. We find that GrAPFI achieves better accuracy and comparable or better coverage. The second part of the dissertation deals with learning representation for biological entities. At the beginning, we focus on neural network-based word embedding technique. We formulate the annotation task as a text classification task. We build a corpus of proteins as sentences composed of respective domains and learn fixed dimensional vector representation for proteins. Then, we focus on learning representation from heterogeneous biological network. We build knowledge graph integrating different sources of information related to proteins and their functions. We formulate the problem of function annotation as a link prediction task between proteins and GO terms. We propose Prot-A-GAN, a machine-learning model inspired by Generative Adversarial Network (GAN) to learn vector representation of biological entities from protein knowledge graph. We observe that Prot-A-GAN works with promising results to associate appropriate functions with query proteins. In conclusion, this thesis revisits the crucial problem of large-scale automatic protein function annotation in the light of innovative techniques of artificial intelligence. It opens up wide perspectives, in particular for the use of knowledge graphs, which are today available in many fields other than protein annotation thanks to the progress of data science.

Keywords: Data Science, Artificial Intelligence,Machine Learning, Big Data, Network Science, Interaction Network, Bioinformatics, Computational Biology, Knowledge Graph, Protein Annotation, Gene Ontology, GO prediction, Enzyme Commission Number, EC prediction, Automatic

Function Annotation, Label Propagation, GrAPFI, Domain Similarity Graph, Representation Learning, Neural Network, Domain Embedding, Sequence Embedding, Heterogeneous Biological Network, Biomedical Knowledge Graph, Knowledge Graph Embedding, Prot-A-GAN, GAN, Generative Adversarial Network for Graph, Protein Annotation GAN