Predicting Protein Functions From Interactions Using Neural Networks and Ontologies

Thesis by

Shahad Qathan

In Partial Fulfillment of the Requirements

For the Degree of

Masters of Science

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

https://orcid.org/0000-0001-2345-6789

ABSTRACT

Predicting Protein Functions From Interactions Using Neural Networks and Ontologies.
Shahad Qathan

To understand the process of life, it is crucial for us to study proteins and their functions. Proteins execute (almost) all cellular activities and their functions are standardized by Gene Ontology (GO). The amount of discovered protein sequences grows rapidly as a consequence of the fast rate of development of technologies in gene sequencing. In UniProKB, there are more than 200 million proteins. Still, less than 1% of the proteins in the UniProtKB database are experimentally GO-annotated, which is the result of the exorbitant cost of biological experiments. To minimize the large gap, developing an efficient and effective method for automatic protein function prediction (AFP) is important.

Many approaches have been proposed to solve the AFP problem, but these methods suffer from limitations in the way the knowledge of the domain is presented and what type of knowledge is included. In this work, we formulate the task of AFP as an entailment problem and exploit the structure of the related knowledge in a set and reusable framework. To achieve this goal, we construct a knowledge base of formal GO axioms and protein-protein interactions to use as background knowledge for AFP. Our experiments show that the approach proposed here, which allows for ontology awareness, improves results for AFP of proteins, they also show the importance of including protein-protein interactions for predicting functions of proteins.

ACKNOWLEDGEMENTS

I want to thank Dr. Robert Hoehndorf, the committee, Dr. Mikhail Moshkov, Dr. Stefan Arold, and the members of the Bio-Ontolgy Research Group for their help and support. Many thanks to my dear family for being there for me during this journey, especially my lovely cat Matas for being kind and understanding about changing my mind on crediting him in my work; I hope this acknowledgment will do. You are a fine specimen of a feline.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AFP	Automatic Function Prediction
BP	Biological Processes
CC	Cellular Component
DLs	Description Logics
GCN	Graph Convolutional Network
GNN	Graph Neural Network
GO	Gene Ontology
KRR	Knowledge Representation and Reasoning
MF	Molecular Function
RGCN	Relational Graph Convolutional Network

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

To understand the process of life, it is crucial for us to study proteins and their functions. Proteins execute (almost) all cellular activities and their functions are standardized by Gene Ontology (GO) with over 44000 classes. The amount of discovered protein sequences grows exponentially due to the fast rate of development technologies in gene sequencing.

Currently, more than 200 million of sequenced proteins are available in the UniProtKB/TrEMB database [1]. Nevertheless, the growth in the number of known sequenced proteins does not extend the biological knowledge we have, because about 1% of the sequenced proteins have a manually-annotated GO function. Additionally, the functional annotation of these sequenced a crucial step for us to understand the biological processes and systems in organisms, it is also one of the most challenging and complex problems in biology, hence why there is a need to develop an efficient and effective an automated protein function prediction (AFP) method.

Many approaches have been proposed for solving the AFP problem. These different strategies can be split into two groups: methods aware of the ontology and methods unaware of the full structure of GO. In this work, we propose a method that aims to some of the main challenges in protein function prediction: predicting functions with strong dependencies in a hierarchical output space, the majority of proteins are not annotated and do not have any known functions, and the challenge of combining information about proteins from different sources. In this approach, the protein function prediction problem is formulated as an entailment problem using Description Logics. The proposed approach allows for

exploitation of both the graph and ontology parts of the knowledge base, thus gaining ontology awareness, to improve the task of protein function prediction, even for classes in GO that only have a few or absolutely no experimental annotations.

Specifically, the contributions in this thesis are as follows:

- Formulate the protein function prediction problem as an entailment problem.
- Construct a knowledge base of formal GO axioms and protein-protein interactions to use as background knowledge for AFP.
- Incorporate graph-based and model-theoretic ontology embedding approaches in the same method.

Chapter 2

Background

The ability to compactly describe data in a specific domain of interest whilst also extracting implicit information of this representation has been one of the main goals of Artificial Intelligence research since its inception. This area of Artificial Intelligence research is called Knowledge Representation and Reasoning (KRR)

Description Logics (DLs) appeared within KRR research by expanding on the taxonomic/structured arrangement of a terminology of a specific domain of interest and giving it a clear, logic-based semantics, first adopted from early network-based approaches, where concepts or classes are interpreted as unary predicates, and relations connecting classes as binary predicates, and more complex expressions can be expressed with logic based constructors inductively. In order for these concepts to be understood, constrains need to be stated in DLs. These constraints can also be utilized to infer their consequences. This is advantageous since the problem is constrained by such knowledge, which can also narrow the search space.

Ontologies, which can be defined as explicit and formal specifications of domains of interest, are comprised of terms expressing precisely defined entities and their interactions. Ontologies are increasingly utilized to define the fundamental concepts and relationships in biological domains, frequently as the basis for data search, integration, and exchange.

Ontologies are utilized as background knowledge in the life sciences since they contain significant information on the entities of the domain, their relationships, and even textual data as comments or definitions. Moreover, ontologies contain information in the form of logical axioms. Such axioms define the domain in

a formal manner. Methods that utilize ontologies exploit their information by acquiring embeddings of the entities of interest.

Gene Ontology (GO) [2] is one of biology's most prominent ontologies. It is used to characterize the genes/proteins of several species by providing concepts/-classes for describing gene activities and their interactions. The objective of this ontology is to transfer the knowledge from well-studied organisms to the lesser or even inaccessible organisms. GO is made up of three primary sub-ontologies: Cellular Component (CC) and Molecular Function (MF) and Biological Processes (BP). CC ontology contains terminology for describing active gene product locations. MF is intended to describe gene product activities. And the BP ontology comprises gene and gene product-contributing processes.

Chapter 3

Related Work

Proteins are the fundamental components of all biological systems, and in order to comprehend the biological system and its molecular behavior, it is vital to comprehend the activities of the proteins. The Gene Ontology (GO) [2] is the most extensive and commonly used database for protein function annotations. It consists of approximately 44,000 classes and a huge number of formal axioms. In the vast, complex, imbalanced, and hierarchical space of biological functions, it continues to be challenging to predict all the true protein functions functions using GO.

Many methods have been used to solve the protein function prediction problem. These methods can be split into two groups: methods aware of the full structure of GO and methods unaware of the GO hierarchy and their other dependencies which predict protein functions flatly.

3.1 AFP with no ontology awareness

DeepGOPlus [3] predicts functions of proteins from sequence alone by incorporating Convolutional Neural Networks, which is a deep learning neural network used for processing structured data, with predictions based on sequence similarity. The CNN model they use analyzes motifs sequences, which are predictive for protein functions, then incorporates it with the functions of related or similar proteins if any are known.

DeepGraphGO is an end-to-end technique for AFP based on graph neural networks. DeepGraphGO tackles two major limitations: a predictive model has to be trained for each type of species, and that the sequences of proteins, which

offers so much valuable information about what the protein does, is completely disregarded in such models. This method overcomes these constraints by proposing a multispecies graph neural network-based solution for AFP that exploits sequences of proteins and complex protein-protein interaction information. Their multispecies approach permits the training of a single model for all species, showing a greater number of training examples than prior methods.

In summary, DeepGraphGO has three notable characteristics: (i) the use of InterPro for vector representation: The input of GNN-trained representation vectors (of nodes/proteins) is derived from InterPro, a database of protein domain and family information [4]. It incorporates 14 distinct databases, such as SUPERFAMILY [5], , Pfam [6] CATH-Gene3D [7] and CDD [8], which contain numerous forms of functional information, including family, domain, and motif. (ii) Multiple Graph Convolutional Network (GCN) layers: GNN has been developed for various purposes, including node classification, link prediction, node embedding, and graph classification [9]. Graph Convolutional Network (GCN) is an example of a classic GNN. By aggregating the representations of neighboring nodes, a graph convolutional layer (GCN layer) may generate a representation vector for each node. Multiple GCN layers permit the capturing of complex ,high-order data between nodes (proteins). (iii) Multispecies strategy: proteins from all species were used to train a single model; this is referred to as the multispecies strategy. Compared to prior work that focused on a single species, this method can utilize more data to achieve higher performance, particularly for species with sparse annotations.

3.2 AFP with ontology awareness

An example of a model that has ontology awareness is DeepGOZero [10], which seeks to improve the prediction of protein functions through the use of the background information included in the Description Logic axioms of GO. Their claim is that ontology axioms would add valuable information and allow for prediction

of functional annotations for ontology terms without training samples (zero-shot) by merging neural and symbolic AI approaches into a single model [11]. GO classes are formally-constrained by GO axioms [2]

DeepGOZero starts by utilizing EL Embeddings, which use the geometric ontology embedding method [12] to generate a n-dimensional space in which GO classes are n-spheres, and the size and location of the n-spheres is constrained by the axioms in GO. Only during the training phase of the model are the ontology axioms used to build the space constrained by axioms of GO. Then, a neural network is employed to project proteins in the same n-dimensional space in which the GO classes were projected to predict the functions of proteins based on how close they are and relation to GO classes. as input, DeepGOZero uses a binary vector of InterPro annotations [4] for proteins. the Multi-layer Perceptron layers transform the interPro annotations binary vector into an embedding vector. Next, DeepGOZero reduces both the protein functions' prediction loss and EL Embeddings loss which constrain the classes. The main limitation with DeepGOZero is that it does not incorporate any network information, thus it suffers from lacking the systems biology perspective.

Chapter 4

Materials and Methods

4.1 Problem formulation

Proteins catalyze metabolic events, replicate DNA, respond to stimuli, provide cells and organisms structure, transport chemicals from one area to another, and many other important functions in all organisms. Which is why understanding protein function is a fundamental problem in biology. Despite its importance, the number of annotated protein functions falls behind* compared to the rapid increase of newly discovered protein sequences. For example, the UniProt database, the most exhaustive collection of functional annotations and protein sequence, has only $< 1\%$ of its proteins annotated.

Gene Ontology (GO), a vast ontology containing more than 44,000 classifications of protein functions, is used to define the protein functions through the ontology classes. It includes the Biological Processes Ontology, the Cellular Component Ontology, and the Molecular Function Ontology. The relationships between the classes within those three sub-ontologies of GO are formally established and must be considered when utilizing them in prediction tasks. Many of the classes of GO have not yet been assigned to any protein, and very few proteins have been assigned a specific function (less than $< 1\%$). This makes the task of training a machine learning model to predict functions with the few or complete absence of annotations very difficult.

There are two main challenges in predicting protein functions, first one is building an efficient computational model that is good at utilizing different types of protein-related knowledge, such as the protein structure, sequence, or even

protein-protein interactions. The other challenge is to predict the right set of functions in the complex, vast, hierarchical and unbalanced space of protein functions as described by GO [10].

Typically, a protein participates in various biological processes, carries out various functions, and is annotated with one or more GO classes concurrently. Therefore, the task of predicting protein function can be viewed as a multi-class, multi-label learning task. Existing multi-label multi-class function prediction methods suffer from the problem of insufficient annotations and a huge number of candidate GO classes due to a large amount of unverified GO annotation of proteins.

4.1.1 Basic definitions

I firstly formulate the protein function predication problem as a logic problem using description logic, then solve it in the embedding space. For this onotology, in order to present proteins, their functions, and interactions. We propose combining GO axioms with knowledge about proteins, their interactions, and their annotations extracted different data sources. The main goal of this is to find the best way to represent this knowledge which covers an arguably large domain.

I adopted the following representation: let O = (C, I, R, Ax) be an ontology, where:

- The set C represents classes in the ontology. I choose all 43329 GO classes for this set.
- the set I represents instances. I choose all 67592464 proteins on STRING Database including all 14094 organisms.
- the set R represent the types of relations that exist in O. The first 6 are relations that may occur between instances of proteins, and the last 2 occur between GO classes. {activates, reacts, binds, expressed by, catalysis, ptmod, regulates, part_of }

- the set Ax includes the following types of axioms:

  1. ?GO_term[1] $\sqsubseteq$?GO_term
  2. ?GO_term $\sqsubseteq \exists$ regulates.?GO_term
  3. ?GO_term $\sqsubseteq \exists$ part_of.?GO_term
  4. ($\exists$ located_in. ?CC) (a)
  5. ($\exists$ has_function. ?MF) (a)
  6. ($\exists$ participates_in. ?BP) (a)
  7. interacts_with[2] (a, b)

The knowledge base of O consists of the TBox and Abox. the TBox is represented by the first 3 axioms, which are generalized concept assertion axioms downloaded from GO released on 2021-11-16. The following 3 are concept assertion axioms extracted from GO annotated proteins sourced from the UniProt/SwissProt Knowledgebase (UniProtKB-SwissProt)[1] version 2022_02. The UniProtKB-SwissProt database contains proteins from approximately 2,000 distinct species. These 3 axioms also represent what we want to predict in the context of protein function prediction and what connects the TBox and ABox in O. The last axiom is of the role assertion type, which represents the ABox and is sourced from STRING Database version 9.1 [13].

4.2 Embedding ontologies

Representation learning approaches are used to account for all knowledge stored in ontologies by classes in the ontology to a continuous representation, also known as an embedding. This is done so that the ontology is embedded in a continues space that allows the execution of computational operations on the ontology information in distributed vector space.

[1] ?GO_term can be either ?CC_GO, ?BP_GO, or ?MF_GO
[2] interacts_with can be any of the relations in the set R

An embedding is a mapping that preserves the structure of one mathematical system to another. In this context, embeddings accept items (individuals, classes, properties) from an ontology and produce a numerical representation in $\mathbb{R}^n$. The size of the embedding is denoted by n. A formal definition of an embedding is:

Definition 1. *Let* $O = (\Sigma = (I, R, C); ax; \vdash)$ *be an ontology with* I *as a set of instances,* R *as a set of relations,* C *as a set of classes,* ax *as a set of axioms and an inference relation* $\vdash$*. the following function is an ontology embedding* $f_\eta : I \cup R \cup C \mapsto \mathbb{R}^n$.

Methods to embed ontologies can be categorized into two main types depending on the type of information from the ontology is utilized; graph-based methods, which are best suited for exploiting the ABox part of the knowledge base. And semantic-based models, which are better at exploiting the TBox part of the knowledge base.

- **Graph-based embedding methods** allow capturing of the structural information of the ontology entities, such as Translational embeddings methods, like TransE, TransC and GCN, which can preserve the graph structure directly via vector operations, but they cannot always represent axioms like symmetry, reflexivity, or transitivity of relations.

  The main challenge in representing axioms graphically is that it is just not possible for all types of axioms, which leads to information loss, this is the major limitation in the graph-based model.

  TransE considers a relation, denoted as r, as a translation from head, denoted as h, to tail, denoted as t, for a triple (h, r, t) in the set of training triples. The embedding vector satisfies the following: h + r $\approx$ t. Consequently, t ought to be the closest neighbor of r+h, while the loss function is calculated as

$$f_r(h, t) = \|h + r - t\|_2^2 \tag{4.1}$$

This graph-based method is suitable for 1-to-1 relations, but cannot handle N-to-1, 1-to-N, or N-to-N types of relations.

TransC [14] instead of embedding classes as points, TransC embeds classes as regions within ℝn. The loss functions in TransC are designed to preserve the is_a transitivity. The formulations of their method is described in the orginal work as follows:

> **Definition 2.** *A Knowledge Graph ($\mathcal{KG}$) describes instances, relations between instances, and classes. It can be expressed as $\mathcal{KG} = \{I, C, R, S\}$. I and C denote the sets of instances and classes respectively. R, the set of relations, can be formalized as $R = \{r_e, r_c\} \cup R_l$, where r_e is an instanceOf relation, r_c is a subClassOf relation, and R_l is the instance relation set.*

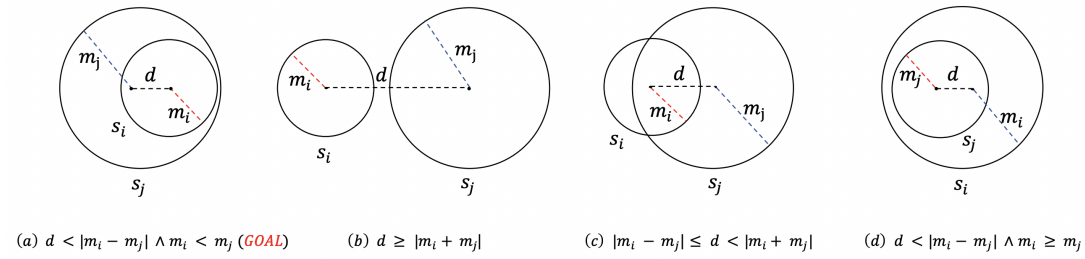

Figure 4.1: Four relative positions between sphere s_i and s_j .

Given knowledge graph $\mathcal{KG}$, knowledge graph embedding with concepts and instances aims at learning embeddings for instances, concepts, and relations in the same space $\mathcal{R}^n$. For each concept c $\in \mathcal{C}$, a sphere s(p, m) is learned with p $\in \mathcal{R}^n$ and m denoting the sphere center and radius. For each instance i $\in \mathcal{I}$ and instance relation r $\in \mathcal{R}^n$ a low-dimensional vector is learned with i $\in \mathcal{R}^n$ and r $\in \mathcal{R}^n$ respectively. Specifically, the instanceOf and subClassOf representations are well-designed so that the transitivity

of isA relations can be reserved.

for each type of the 3 triples, the loss function is calculated as follows:

– InstanceOf Triple Representation. For a given instanceOf triple (i, r_e, c), if it is a true triple, i should be inside the sphere s to represent the instanceOf relation between them. If i is outside the sphere s, the embeddings would need to be optimized. The loss function that achieves this is defined as

$$f_e(i, c) = ||i - p||_2 - m. \tag{4.2}$$

– Relational Triple Representation. For a relational triple (h, r, t), TransC will learn low dimensional vectors h, t, r $\in \mathcal{R}^n$ for instances and relations. Just like TransE (Bordes et al., 2013), the loss function of this kind of triples is defined as

$$f_r(h, t) = ||h + r - t||_2^2 \tag{4.3}$$

– SubClassOf Triple Representation. For a subClassOf triple (c_i, r_c, c_j) concepts c_i, c_j are encoded as spheres $s_i(p_i, m_i)$ and $s_j(p_j, m_j)$. The distance between the centers of the two spheres is first calculated as

$$d = ||p_i - p_j||_2 \tag{4.4}$$

If (c_i, r_c, c_j) is a true triple, sphere s_i should be inside sphere s_j Figure 5.1a to represent the subClassOf relation between them. Actually, there are three other relative positions between sphere s_i and s_j (as shown in Figure 5.1). We also have three loss functions under these three conditions:

1. s_i is separate from s_j (Figure 5.1b). The embeddings still need to be optimized. In this condition, the two spheres need to get closer in optimization. Therefore, the loss function is defined as

$$f_c(c_i, c_j) = ||p_i - p_j||_2 + m_i - m_j. \tag{4.5}$$

– s_i intersects with s_j (Figure 5.1c). This condition is similar to condition 1. The loss function is defined as

$$f_c(c_i, c_j) = ||p_i - p_j||_2 + m_i - m_j. \tag{4.6}$$

– s_j is inside s_i (Figure 5.1d). It is different from the target and m_j should be reduced with m_i increased. Hence, the loss function is

$$f_c(c_i, c_j) = m_i - m_j[14] \tag{4.7}$$

Graph Convolutional Networks

The Graph Convolutional Networks (GCN) methods define the convolution operation of the graph by message passing through the its edges. They are heavily influenced by convolutional neural networks, which consider the spatial structure of the input grid-structured data. CNNs can only operate on Euclidean structured data, whereas GCNs are a generalized version of CNNs in which the number of node connections varies and the nodes are not ordered.

The GCN [15] is type of layer which inputs a set of vectors that represent vertices, Incorporated with the structure of the full graph and outputs new set of representations for vertices of the graph. A directed graph is denoted as $\mathcal{G} = (\mathcal{V}; \mathcal{E})$, with set $\mathcal{V}$ being of vertices of the graph and $\langle i, i \rangle \in \mathcal{E}$ being

the set of directed edges between two vertices, pointing from vertex i to j. The following equation demonstrates the message passing of GCN for an undirected graph in a single layer, $\mathcal{G}$.

$$H = \sigma(AXW) \tag{4.8}$$

Where X represents the matrix of vertex features, W is the matrix of weight parameters, and σ is the non-linear activation function. A is generated by the normalization of the adjacency matrix of the graph $\mathcal{G}$. Normalization guarantees that the magnitude of the vectors of vertex features does not significantly change during the message-passing step.

Relational Convolutional Networks are an extension of GCN which is used for multi-relational graphs. They allow for the basic message passing to be extended to Knowledge Graphs (KG) By taking into account each edge' direction and handling message-passing for each relation independently. A KG, which is a directed graph, is denoted as $\mathcal{G} = (\mathcal{V}; \mathcal{E}; \mathcal{R})$, where $\mathcal{R}$ represents the set of relations and $\langle s, r, o \rangle \in \mathcal{E}$ is a set of triples representing that an instance (head) s and an instance (tail) o are connected by relation r $\in \mathcal{R}$. The following equation is an extension of the GNN message passing rule as descibed in the original paper [16] and in [17].

$$H = \sigma\Big(\sum_{r=1}^{R} A_r X W_r\Big) \tag{4.9}$$

where R is the number of relations, A_r is an adjacency matrix describing the edge connection for a given relation r and W_r is a relation-specific weight matrix. The extended message passing rule denotes how the information should be mixed together with neighboring nodes in a relational graph. In the message passing step, the embedding is summed over the different relations. With the message passing rule discussed thus far, the

problem is that for a given triple $\langle s, r, o \rangle$ a message is passed from s to o, but not from o to s. For instance, for the triple $\langle protein_1, interacts_with, protein_2 \rangle$ it would be desirable to update both *protein_1* with information from The *protein_2*, and *protein_2* with information from *protein_1*, while modelling the two directions as meaning different things. To allow the model to pass messages in two directions, the graph is amended inside the RGCN layer by including inverse edges: for each existing edge $\langle s, r, o \rangle$ a new edge $\langle o, r', s \rangle$ is added where r' is a new relation representing the inverse of r. A second problem with the naive implementation of the $(R)GCN$ is that the output representation for a node i does not retain any of the information from the input representation. To allow such information to be retained, a self-loop $\langle s, r_s, s \rangle$ is added to each node, where r_s is a new relation that expresses identity. Altogether, if the input graph contains R relations, the amended graph contains $2R + 1$ relations: $R^+ = R \cup R' \cup R_s$

- The second approach we are considering for generating embeddings is a model-semantic approach. This type of method represents the actual interpretation of the symbols and exploit the TBox part of the knowledge base. An example of a method of this type is **EL Embeddings**, which aim to embed ontologies as n-sphers, and the relationships between entities is represented by relations of intersection or inclusion between those spheres. The binary crossentropy loss between the true labels and predicted labels is calculated, then optimized together with four normal form losses for ontology axioms from EL Embeddings. Adam optimizer [18] is used to minimize the following loss function as described in the orginial work [12]:

$$L = \frac{1}{N}\sum_{i=1}^{N} BCELoss(y_{c_i}, y'_{c_i}) + L_{NF1} + L_{NF2} + L_{NF3} + L_{NF4} \tag{4.10}$$

EL Embeddings use normalized axioms in the following four different forms:

- NF1 : C $\sqsubseteq$ D
- NF2 : $C \sqcap D \sqsubseteq$ E
- NF3 : C $\sqsubseteq \exists$ R.D
- NF4 : $\exists R.C \sqsubseteq$ D

where C, D, E represent classes and R represents relations in the ontology. We convert the GO axioms into these four normal forms using a set of conversion rules

4.3 Overview

Figure 4.2 shows a high-level overview of our proposed method. The goal of this approach is to predict protein functions while utilizing the axioms in GO. Specifically, a n-dimensional space is generated where go classes are represented as n-spheres, having the size and location of each sphere constrained by axioms in GO. Then, GCN is used to project proteins in the same n-dimensional space where GO classes are embedded. Function prediction will depend on proteins and their relations and proximity to GO classes. In the proposed method, InterPro annotations [4] are used as feature vectors for each protein. The binary vector of InterPro domain annotations is processed by the GCN layer to generate an embedding vector. Then, it minimizes the loss of protein function prections and the EL Embeddings/TransC loss at the same time.

4.4 Performance evaluation

the model performance is evaluated using the measurements previously used in CAFA challenge [19]. The first performance measurement is a protein centric F-measure. Where the F-measure is calculated for a threshold t ∈ [0, 1] with the averaged precision for proteins that have at least one predicted term and also the averaged recall for all proteins. Next, the maximum F-measure of all threshold values between [0, 1] is selected. F_{max} measure is computed with the following formulas:

$$pr_i(t) = \frac{\sum_f I(f \in P_i(t) \wedge f \in T_i)}{\sum_f I(f \in P_i(t))} \tag{4.11}$$

$$rc_i(t) = \frac{\sum_f I(f \in P_i(t) \wedge f \in T_i))}{\sum_f I(f \in T_i)} \tag{4.12}$$

$$AvgPr(t) = \frac{1}{m(t)} \cdot \sum_{i=1}^{m} (t) pr_i(t) \tag{4.13}$$

$$AvgRc(t) = \frac{1}{n} \cdot \sum_{i=1}^{n} rc_i(t) \tag{4.14}$$

$$F_{max} = \max_t \left\{ \frac{2 \cdot AvgPr(t) \cdot AvgRc(t)}{AvgPr(t) + AvgRc(t)} \right\} \tag{4.15}$$

where f is GO class, Pi(t) is a set of predicted classes for a protein i using a threshold t, and Ti is a set of annotated classes for a protein i. The average precision is computed over the proteins that got one or more predicted terms, m(t) represents the number of such proteins. and n represents the number of the total proteins in the test set.

S_{min} computes the semantic distance between predicted and real annotations based on information content (IC) of the classes. The IC(c) is computed using the class c's label probability:

$$IC(c) = -\log(Pr(c|P(c))) \tag{4.16}$$

where P(c) is a set of parent classes of the class c. The S_{min} is computed using the following formulas:

$$S_{min} = min_t \sqrt{ru(t)^2 + mi(t)^2} \tag{4.17}$$

where mi(t) is average amount of misinformation and ru(t) is the average remaining uncertainty :

$$ru(t) = \frac{1}{n} \sum_{i=1}^{n} \sum_{c \in T_i - P_i(t)} IC(c) \tag{4.18}$$

$$mi(t) = \frac{1}{n} \sum_{i=1}^{n} \sum_{c \in P_i(t) - T_i} IC(c) \tag{4.19}$$

4.5 Model training settings

To train our prediction models on this dataset and ensure that it generalizes well to novel proteins, we use the same split used in DeepGOZero [10]. The split is done before the generation of the knowledge base, this is because we want to measure how well the method is at predicting proteins and GO terms it has never came across in training. The split is described in the orginial work as follows:

> we split the proteins into training, validation and testing sets. If we split proteins randomly, the generated dataset splits will contain proteins that are very similar or almost identical. Using such datasets in machine learning may lead to an overfitting problem and the prediction models will not generalize well (Tetko et al., 1995). Therefore, we group the proteins by their similarity before generating a random split. We place proteins with sequence identity over 50% into the same groups and used 81% of the groups for training, 9% for validation and 10% for testing.

Also, each of the GO sub-ontologies is trained and evaluated separately, and the training utilizes data of all species (multispecies). Additionally, mini-batch training is used, since GraphSAGE showed a better generalized performance [20], instead of full-batch training in [15]. The effectiveness of mini-batch training was also demonstrated in DeepGraphGO [21]

4.6 Complexity

The complexity of training and generating the embeddings of proteins using GCNs is calculated as $O(N * F^2 + N^2 * F)$ [22], while the training time of the GO axiom embeddings is linear O(epochs * n * m), where n is the number of all axioms, m is an embedding size and epochs is the number of training iterations. And the last step, which is generating inferences, takes O(embedding size) time, thus it is linear in the number of all entities.

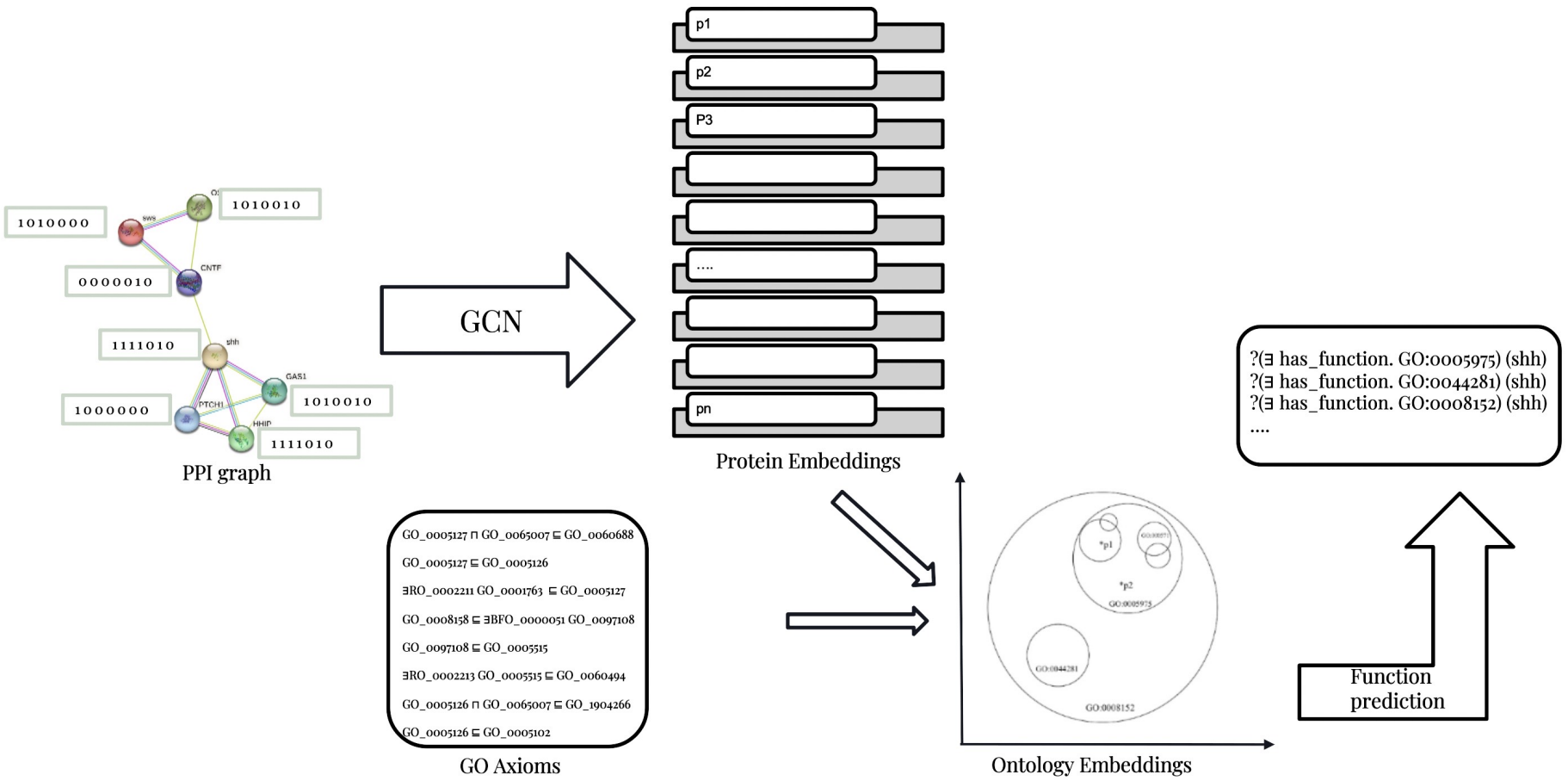

Figure 4.2: high-level overview of our proposed method.

Chapter 5

Results and Discussion

The aim of our proposed approach is to utilizes the knowledge embeded in GO axioms and protein-protein interactions to improve protein function prediction. This is achieved through jointly generating protein and GO embeddings in the same continuous space. The hypothesis is that the GO axioms will help improve the predictions quality and allow for prediction of protein functions for GO classes whcih do not have any training samples (zero-shot), combining symbolic and neural AI methods in a single model [23]. We also include protein interactions because we know proteins do not work in isolation thus this type of knowledge is crucial especially for two specific domains of GO: the Biological Processes, because BP terms are used to describe sets of molecular events with a defined beginning and end, which implies the involvement of multiple (*interacting*) proteins, and Cellular Component, which describes the parts of a cell or its extracellular environment where a protein is, and by intuition, proteins in the same location in the cell are likely to interact.

The goal of incorporating GCN with other ontology embeddings methods (TransC and EL Embeddings in the following expermints) is to merge benefits of GCN in embedding graphs in $\mathcal{R}^n$ which is suitable for the ABox part of the ontology, and the ability of TransC/EL Embeddings to of embed the GO axioms, the TBox part of the ontology.

Table 5.1 shows the first set of experiments where regular graph-based embedding methods (TransC, GCN) are compared against our proposed approach (GCN+TransC), which has showed an improvement compared to solely using TransC due to GCNs having the ability to embed protein sequence features, the

protein features were omitted in TransC because the method does not allow for node features. In comparison with GCN, GCN+TransC shows minor improvements in MF and CC domains, this is due to the fact that our method allows for GO embeddings while a GCN does not.

in Table 5.2, we compare our proposed methods with current state of the art DeepGraphGO [21]. The experiments show some improvements in MF and CC domains, which shows that ontology awareness improves AFP of proteins. The combination of GCN and EL Embeddings performs better than GCN with TransC; this is due to the fact that EL Embeddings is more suited to embed the TBox part of the knowledge base while GCN embeds the ABox, while in GCN+TransC, TransC is not as suited for embedding TBoxes.

The second set of experiments against current state of the art compares our methods with DeepGOZero [10] in Table 5.3, which is an ontology aware method. The results show better performance in all domains. This shows that the incorporation of protein interactions knowledge improves performance for AFP of proteins.

Moreover, all models perform significantly worse in the BP sub-ontology than in the MF and CC sub-ontologies This is consistent with the CAFA findings, which may be attributed to the following factors mentioned in [21]:

- BP has much more GO terms and higher depths than MF and CC;
- the BP terms are considered to be more abstract in nature than MF and CC terms;
- BP may have complicated annotation status such as the annotation depth of benchmark proteins and various annotation biases.

Method	F-max			S-min			AURP		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
TransC	0.535	0.431	0.631	12.459	45.699	11.021	0.524	0.398	0.650
GCN	0.590	**0.467**	0.663	**11.074**	**43.303**	10.582	0.597	**0.442**	0.675
RGCN+ TransC	**0.595**	0.442	**0.685**	11.590	45.198	**9.395**	**0.599**	0.424	**0.701**

Table 5.1: Comparison of performance against graph-based embedding approaches

Method	F-max			S-min			AURP		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
DGG	0.590	**0.467**	0.663	11.074	**43.303**	10.582	0.597	**0.442**	0.675
RGCN+ TransC	0.595	0.442	0.685	11.590	45.198	**9.395**	0.599	0.424	0.701
RGCN+ EL emb	**0.641**	0.459	**0.688**	**10.148**	45.091	11.183	**0.652**	0.433	**0.721**

Table 5.2: Comparison of proposed methods against DeepGraphGO

Method	F-max			S-min			AURP		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
DGZ	0.635	0.446	0.662	11.748	51.413	10.617	0.625	0.414	0.653
RGCN+ TransC	0.595	0.442	0.685	11.590	45.198	**9.395**	0.599	0.424	0.701
RGCN+ EL emb	**0.641**	**0.459**	**0.688**	**10.148**	**45.091**	11.183	**0.652**	**0.433**	**0.721**

Table 5.3: Comparison of proposed methods against DeepGOZero

Method	F-max			S-min			AURP		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
GCN	0.590	0.431	0.631	11.748	51.413	10.617	**0.625**	0.414	0.653
RGCN	**0.595**	**0.442**	**0.685**	**11.590**	**45.198**	**9.395**	0.599	**0.424**	**0.701**

Table 5.4: Comparison of performance of GCN and RGCN in proposed method

5.1 Hyperparameters analysis

For hyperparameter tuning, we test for the appropriate number of embedding dimensions, ranging from 64 up to 2024 dimensions. The performance was improved slightly with the increase in dimensions but the space complexity suffered. MF and BP also show a stall in perfomance after 1024 dimensions.

We also compared the use of a regular GCN layer against a RGCN layer in our approach, where in the RGCN, for each type of interaction between proteins, a different weight matrix would be considered. This showed some improvement compared to GCN which only considers one type of relation.

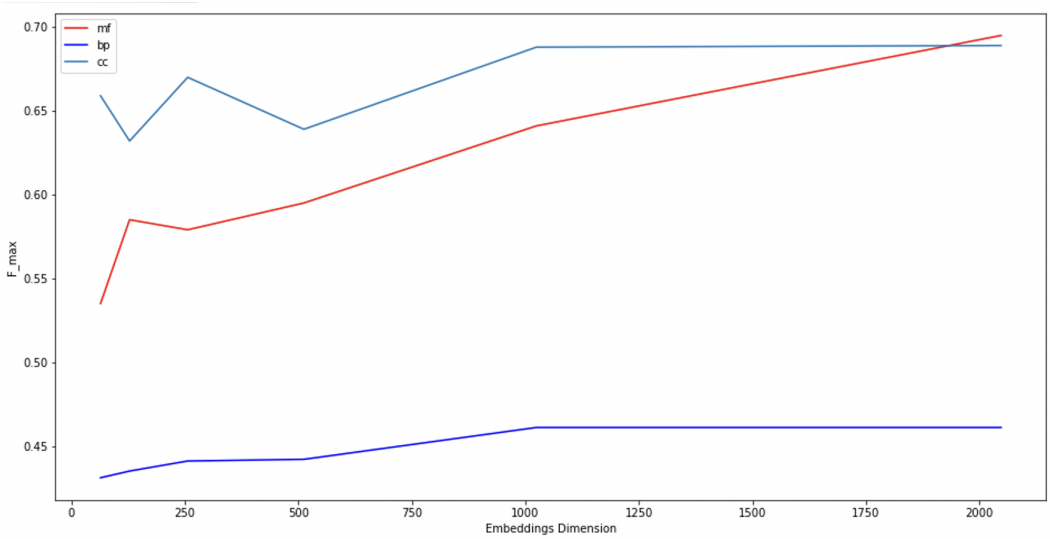

Figure 5.1: The performance slightly improves after increasing embedding dimensions

Chapter 6

Concluding Remarks

To understand the process of life, it is crucial for us to study proteins and their functions. Proteins execute (almost) all cellular activities and their functions are standardized by Gene Ontology (GO). The amount of discovered sequences of proteins grows exponentially as a consequence of the fast rate of development of technologies in gene sequencing. To reduce this wide gap, developing an efficient and effective method for automatic protein function prediction (AFP) is important. Many approaches have been proposed for solving the AFP problem, but these methods suffer from limitations in the way the knowledge of the domain is presented. In this work, I formulate the task of AFP as an entailment problem and exploit the structure of the related knowledge in a set and reusable framework. We aimed to answer some research questions including: How does ontology awareness affect performance of AFP? does inclusion of protein interaction knowledge improve performance of AFP? and what is the best way to combine ontology embedding methods suited for the problem of protein function prediction?.

Our experiments show that ontology awareness improves protein function prediction, and combining an embedding method that is suited for the graph structure (ABox) and the ontology structure (TBox) of the knowledge base yields the best results. We also demonstrated how protein-protein interactions add valuable knowledge that allows for better performance overall. To improve our method, more experiments considering different protein knowledge, such as protein structure, should be considered, and even implementing other Deep Learning architectures. The main limitation we faced was that both ontology embedding

methods we experminted with (TransC, EL Embeddings) were similar due to the fact that they are both translational approaches. In future work, we will consider different methods that are better suited for this large-domain problem.

REFERENCES

[1] R. Apweiler, A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane *et al.*, "Uniprot: the universal protein knowledgebase," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D115–D119, 2004.

[2] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig *et al.*, "Gene ontology: tool for the unification of biology," *Nature genetics*, vol. 25, no. 1, pp. 25–29, 2000.

[3] M. Kulmanov and R. Hoehndorf, "Deepgoplus: improved protein function prediction from sequence," *Bioinformatics*, vol. 36, no. 2, pp. 422–429, 2020.

[4] A. L. Mitchell, T. K. Attwood, P. C. Babbitt, M. Blum, P. Bork, A. Bridge, S. D. Brown, H.-Y. Chang, S. El-Gebali, M. I. Fraser *et al.*, "Interpro in 2019: improving coverage, classification and access to protein sequence annotations," *Nucleic acids research*, vol. 47, no. D1, pp. D351–D360, 2019.

[5] M. E. Oates, J. Stahlhacke, D. V. Vavoulis, B. Smithers, O. J. Rackham, A. J. Sardar, J. Zaucha, N. Thurlby, H. Fang, and J. Gough, "The superfamily 1.75 database in 2014: a doubling of data," *Nucleic acids research*, vol. 43, no. D1, pp. D227–D233, 2015.

[6] R. D. Finn, P. Coggill, R. Y. Eberhardt, S. R. Eddy, J. Mistry, A. L. Mitchell, S. C. Potter, M. Punta, M. Qureshi, A. Sangrador-Vegas *et al.*, "The pfam protein families database: towards a more sustainable future," *Nucleic acids research*, vol. 44, no. D1, pp. D279–D285, 2016.

[7] T. E. Lewis, I. Sillitoe, N. Dawson, S. D. Lam, T. Clarke, D. Lee, C. Orengo, and J. Lees, "Gene3d: extensive prediction of globular domains in proteins," *Nucleic acids research*, vol. 46, no. D1, pp. D435–D439, 2018.

[8] A. Marchler-Bauer, Y. Bo, L. Han, J. He, C. J. Lanczycki, S. Lu, F. Chitsaz, M. K. Derbyshire, R. C. Geer, N. R. Gonzales *et al.*, "Cdd/sparcle: functional classification of proteins via subfamily domain architectures," *Nucleic acids research*, vol. 45, no. D1, pp. D200–D203, 2017.

[9] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," 2018. [Online]. Available: https://arxiv.org/abs/1812.08434

[10] M. Kulmanov and R. Hoehndorf, "Deepgozero: Improving protein function prediction from sequence and zero-shot learning based on ontology axioms," *bioRxiv*, 2022.

[11] J. Mira and A. Delgado, "Where is knowledge in robotics? some methodological issues on symbolic and connectionist perspectives of ai," in *Autonomous robotic systems*. Springer, 2003, pp. 3–34.

[12] M. Kulmanov, W. Liu-Wei, Y. Yan, and R. Hoehndorf, "El embeddings: geometric construction of models for the description logic el++," *arXiv preprint arXiv:1902.10499*, 2019.

[13] A. Franceschini, D. Szklarczyk, S. Frankild, M. Kuhn, M. Simonovic, A. Roth, J. Lin, P. Minguez, P. Bork, C. Von Mering *et al.*, "String v9. 1: protein-protein interaction networks, with increased coverage and integration," *Nucleic acids research*, vol. 41, no. D1, pp. D808–D815, 2012.

[14] X. Lv, L. Hou, J. Li, and Z. Liu, "Differentiating concepts and instances for knowledge graph embedding," 2018. [Online]. Available: https://arxiv.org/abs/1811.04588

[15] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[16] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[17] T. Thanapalasingam, L. van Berkel, P. Bloem, and P. Groth, "Relational graph convolutional networks: a closer look," *PeerJ Computer Science*, vol. 8, p. e1073, 2022.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur *et al.*, "A large-scale evaluation of computational protein function prediction," *Nature methods*, vol. 10, no. 3, pp. 221–227, 2013.

[20] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[21] R. You, S. Yao, H. Mamitsuka, and S. Zhu, "Deepgraphgo: graph neural network for large-scale, multispecies protein function prediction," *Bioinformatics*, vol. 37, no. Supplement_1, pp. i262–i271, 2021.

[22] D. Blakely, J. Lanchantin, and Y. Qi, "Time and space complexity of graph convolutional networks," *Accessed on: Dec*, vol. 31, 2021.

[23] J. Mira, A. Delgado, and M. Taboada, "Neurosymbolic integration: The knowledge level approach," in *International Conference on Computer Aided Systems Theory*. Springer, 2003, pp. 460–470.