PREDICTING INTRACELLULAR FUNCTIONS OF PROTEINS FROM AMINO ACID SEQUENCES USING LANGUAGE PROCESSING METHODS

by

Bedirhan Çaldır

B.S., Computer Engineering, Koç University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2022

ACKNOWLEDGEMENTS

I would like to thank my advisor Assoc. Prof. Arzucan Özgür for their guidance, support and patience during my studies.

I would like to express my gratitude to my jury members, Prof. Tunga Güngör and Prof. Attila Gürsoy for kindly accepting to participate in the defense of this thesis.

I would also like to thank the members of TABILAB, especially to Burak Suyunu, Enes Taylan and Gökçe Uludoğan for discussions, small talks and useful advises during this thesis was being prepared from start to end.

I am grateful to my dear wife, M.D. Ayşe Uysal Çaldır, and my family as they have always been supporting and pushing me to be who I am.

Lastly, Atılberk Çelebi and Ceyhun Onur deserve a big appreciation for chasing me and never giving up since our Freshman. I hope it lasts forever.

ABSTRACT

PREDICTING INTRACELLULAR FUNCTIONS OF PROTEINS FROM AMINO ACID SEQUENCES USING LANGUAGE PROCESSING METHODS

Rapidly increasing computational power and sequencing technologies, which are at the peak of their development, enable the use of advanced algorithms with high processing volume to predict the intracellular functions of proteins, which is one of the most important problems in computational biology. The functionalities of proteins emerge primarily through their three-dimensional folded structures. When these structures are interpreted as graphs, the application of graph neural networks leads to promising results. However, these approaches are limited as the three-dimensional folded structures are not yet known for most proteins. The fact that the amino acid sequences of proteins have properties similar to natural languages and the large amounts of sequence data suggest that these sequences can be processed using natural language processing (NLP) methods. In this thesis, two different NLP methods are adapted to the problem of protein function prediction, assuming that the protein sequence data contain necessary and sufficient information to predict both three-dimensional folded structure and intracellular function: (i) Bidirectional Transoformer BERT model (ii) Heterogeneous Graph Convolutional Network (GCN) model. The results show that it is more advantageous to treat the proteins as graphs. The GCN model performs better than the BERT model and achieves performance close to the state-of-the-art model that uses three-dimensional folding information. In addition, we find that tokenizing the sequences instead of using the individual amino acids as tokens increases the performance.

ÖZET

PROTEİNLERİN AMİNO ASİT DİZİLİMLERİNİN DOĞAL DİL İŞLEME METOTLARI İLE İŞLENEREK HÜCRE İÇİ FONKSİYONLARININ TAHMİNLENMESİ

Günümüzde hızla yükselen hesaplama gücü ve gelişiminin doruklarında olan sekanslama teknolojileri hesaplamalı biyoloji alanındaki en önemli sorunlardan olan proteinlerin hücre içi fonksiyonlarının tahminlenebilmesi için yüksek işlem hacmi ile ileri seviye algoritmaların kullanılabilmesine olanak sağlamaktadır. Proteinlerin hücre içi işlevleri aslen üç boyutlu katlanmış yapıları sayesinde ortaya çıkmaktadır. Bu yapılar bir çizge olarak yorumlandığında çizge sinir ağları bu yapılara uygulandığında umut verici sonuçlar ortaya çıkmaktadır. Fakat çoğu protein için üç boyutlu katlanmış yapılar henüz yeterince bilinmediğinden bu yaklaşımlar kısıtlı kalmaktadır. Bununla birlikte proteinlerin amino asit dizilerinin doğal dillere benzer niteliklere sahip olması ve büyük miktarlarda dizi verisi bulunması bu dizilerin doğal dil işleme yöntemleri ile de işlenebileceğini ve işlevsel tahmin yapılabileceğini işaret etmektedir. Bu tezde, protein dizisi verisinin üç boyutlu katlanmış yapıyı da hücre içi fonksiyonu da tahmin etmek için yeterli bilgiyi içerdiği varsayımı ile iki farklı doğal dil işleme yöntemi probleme uyarlanmıştır: (i) çift yönlü dönüştürücü temelli BERT modeli (ii) çizge sinir ağları temelli heterojen çizge evrişimsel sinir ağı modeli. Sonuçlar, proteinleri çizge şeklinde yorumlamanın daha avantajlı olduğunu ortaya koymuştur. Çizge evrişimsel sinir ağları modeli BERT modelinden daha başarılı sonuçlar üretmiş ve üç boyutlu katlanmış yapıların bilgisini de kullanan son teknoloji çizge temelli modele yakın bir performans göstermiştir. Ayrıca, dizideki her amino asidin tek tek kullanılması yerine gruplandırılmış şekilde kullanılmasının daha başarılı sonuçlar ürettiği gözlemlenmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . xi
LIST OF SYMBOLS . . . xii
LIST OF ACRONYMS/ABBREVIATIONS . . . xiii
1. INTRODUCTION . . . 1
2. BACKGROUND . . . 3
2.1. Proteins . . . 3
2.2. Amino Acids . . . 3
2.3. Protein Functionalities . . . 4
2.4. Databases . . . 4
2.4.1. Protein Data Bank (PDB) . . . 4
2.4.2. Structure Integration with Function, Taxonomy and Sequence (SIFTS) . . . 5
2.4.3. UniProtKB/SWISS-Prot . . . 5
2.5. Technical Background . . . 6
2.5.1. Transformers . . . 6
2.5.2. Bidirectional Encoder Representations from Transformers (BERT) 7
2.5.3. Byte-Pair Encoding (BPE) . . . 9
2.5.4. Graph Convolutional Networks (GCN) . . . 11
2.5.5. Point-wise Mutual Information (PMI) . . . 14
2.5.6. Term Frequency-Inverse Document Frequency (TF-IDF) . . . 15
2.6. Related Work . . . 15
3. MATERIALS AND METHODS . . . 20
3.1. Data . . . 20
3.2. Models . . . 27

3.2.1. Prot-BERT and Fine-Tuning 27
3.2.2. Text-GCN 29
3.2.3. Loss Functions 31
3.2.3.1. Binary Cross Entropy Loss 32
3.2.3.2. Focal Loss 33
3.3. Evaluation 34
3.3.1. Benchmarking metrics 34
4. EXPERIMENTS AND RESULTS 36
4.1. Experimental Setup 36
4.2. Model Hyperparameters 37
4.3. Experiments and Results 39
4.3.1. Hyperparameter Analysis 39
4.3.2. Comparison Results 43
5. DISCUSSION 54
6. CONCLUSION 57
REFERENCES 59
APPENDIX A: ADDITIONAL STATISTICS OF DATASETS 66
APPENDIX B: COMPARISONS OF PROT-BERT AND TEXT-GCN 70
APPENDIX C: ADDITIONAL FIGURES OF PROT-BERT 75
APPENDIX D: ADDITIONAL FIGURES OF TEXT-GCN 78

LIST OF FIGURES

Figure 3.1. Histogram of samples in terms of number of functions associated with them for MF class type in PDB dataset. 22

Figure 3.2. Cumulative positive label counts indicating the sparsity of the labels for each dataset. 23

Figure 3.3. Word length distribution of encoded datasets for different BPE tokenization settings. 27

Figure 4.1. The hyperparameters affecting performance the most and their success for different options for Text-GCN. 39

Figure 4.2. The hyperparameters affecting performance the most and their success for different options for Prot-BERT. 40

Figure 4.3. The effect of BPE tokenization in terms of accuracy and F1 scores. 41

Figure 4.4. Text-GCN ROC curves and areas under them for the proteins with PDB-IDs *1CNT-1*, *1DI7-A*, *1E0T-A*, *1QMT-A*, *1UWY-A* and *1ZI7-A*. 46

Figure 4.5. Text-GCN ROC curves and areas under them for the randomly sampled proteins. 47

Figure 4.6. Text-GCN F1 scores of each label in MF ordered by their presence in ascending order. 48

Figure 4.7. Histogram showing the cumulative sequence length distribution. . 50

Figure 4.8. Histogram showing the sequence length distribution that are longer than 512 amino acids (AA). 51

Figure 4.9. Total amino acid count of the samples distributed over their lengths. 52

Figure A.1. Histogram of samples in terms of number of functions associated with them for CC class type in PDB dataset. 66

Figure A.2. Histogram of samples in terms of number of functions associated with them for BP class type in PDB dataset. 67

Figure A.3. The same information with Figure 3.2 but limited at 1000 to focus on the labels occurring rarely. 68

Figure A.4. The same information with Figure 3.2 but limited at 100 to focus on the labels occurring extremely rarely. 69

Figure B.1. Accuracy and F1 score comparison for different loss function options used during model trainings. 70

Figure B.2. Prot-BERT vs Text-GCN ROC curves and area under them for protein with PDB-ID *5F1A-A* where Text-GCN performed better than Prot-BERT. 71

Figure B.3. Prot-BERT vs Text-GCN ROC curves and area under them for protein with PDB-ID *1QPG-A* where Text-GCN performed similarly to Prot-BERT. 72

Figure B.4. Prot-BERT vs Text-GCN ROC curves and area under them for protein with PDB-ID *4PAC-A* where Prot-BERT performed better than Text-GCN which is slightly better than a random classifier. . 73

Figure B.5. Prot-BERT vs Text-GCN ROC curves and area under them for protein with PDB-ID *4PAC-A* where both models performed worse than a random classifier. 74

Figure C.1. Prot-BERT F1 scores of each label in MF ordered by their presence in ascending order. 75

Figure C.2. Prot-BERT F1 scores of each label in BP ordered by their presence in ascending order. 76

Figure C.3. Prot-BERT F1 scores of each label in CC ordered by their presence in ascending order. 77

Figure D.1. Text-GCN F1 scores of each label in BP ordered by their presence in ascending order. 78

Figure D.2. Text-GCN F1 scores of each label in CC ordered by their presence in ascending order. 79

LIST OF TABLES

Table 3.1. Summary statistics of datasets. 21

Table 3.2. Summary document statistics for different BPE tokenization settings. 25

Table 3.3. Summary word statistics for different BPE tokenization settings. . 26

Table 4.1. Specifications of the computer 36

Table 4.2. Programming language, its libraries/frameworks and their versions 36

Table 4.3. Hyperparameters and their options for Prot-BERT 37

Table 4.4. Hyperparameters and their options of Text-GCN 38

Table 4.5. Fixed hyperparameters for the models 42

Table 4.6. Performance of both models and its comparison to the state-of-the-art DeepFRI model along with other competitive models. 43

Table 4.7. Performance of the two proposed models compared to each other for each of the functional class types. 45

Table 4.8. The results of sparsity experiment with two different subsets . . . 49

Table 4.9. Performance of the two proposed models trained and tested with the clipped data for each of the functional class types. 53

LIST OF SYMBOLS

A_{ij}	Adjacency information of two nodes **i** and **j** in a graph
$\mathbf{A}$	Adjacency matrix of a graph neural network
$\mathbf{c_j}$	Class **j**
$\mathbf{D}$	Diagonal node degree matrix of adjacency matrix **A** of a GCN
$\mathbf{H}(.)$	Loss function
$\mathbf{I}$	Identity matrix
$\mathbf{IDF}(\mathbf{w}, \mathbf{C})$	Inverse document frequency of a word **w** in a corpus **C**
$\mathbf{ln}(.)$	Natural logarithm function
$\mathbf{log}(.)$	Logarithm function
$\hat{\mathbf{M}}$	Identity matrix **I** added to matrix **M**
$\mathbf{N}$	Sample size in a dataset
$\mathbf{p_{c_j}}$	Weight of positive samples for class **j**
$\mathbf{P}(\mathbf{i})$	Marginal probability of a word **i**
$\mathbf{P}(\mathbf{i}, \mathbf{j})$	Joint probability of a word pair **i** and **j**
$\mathbf{PMI}(\mathbf{i}, \mathbf{j})$	Point-wise mutual information of a word pair **i** and **j**
$\mathbf{TF}(\mathbf{w}, \mathbf{d})$	Term frequency of a word **w** in a document **d**
$\mathbf{W}$	Weight matrix of a neural network
$\mathbf{x}$	The input features of all samples
$\mathbf{x_j}$	The input features of sample **j**
$\mathbf{y}$	True label of all samples
$\mathbf{y_j}$	True label of sample **j**
$\hat{\mathbf{y}}_\mathbf{j}$	Predicted label of sample **j**
γ	Relaxation parameter for focal loss
$\sigma(.)$	Non-linear function such as Sigmoid, ReLU or Softmax

LIST OF ACRONYMS/ABBREVIATIONS

AUROC	Area Under the Receiver Operating Characteristics
BERT	Bidirectional Encoder Representations from Transformers
BP	Biological Processes
BPE	Byte-Pair Encoding
CAFA	Critical Assessment of Functional Annotation
CC	Cellular Component
CNN	Convolutional Neural Network
FPR	False Positive Rate
GCN	Graph Convolutional Network
GO	Gene Ontology
K	Thousand
LM	Language Model
LR	Learning Rate
LSTM	Long Short-Term Memory
M	Million
MF	Molecular Function
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
PDB	Protein Data Bank
PMI	Point-wise Mutual Information
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SIFTS	Structure Integration with Function, Taxonomy, Sequence
TF-IDF	Term Frequency-Inverse Document Frequency
TPR	True Positive Rate
UniProt	The Universal Protein Resource

1. INTRODUCTION

In the context of modern biology, proteins and their functionalities have become more important than ever because it is understood better that they play a crucial role for cells to be "alive". They are a type of macromolecule used by cells to perform a variety of tasks in the cell. They aid in metabolism by serving as enzymes, hormones and transporters, as well as providing structural support to the cell. However, proteins with well-defined functions make up only a tiny proportion of the characterized and known proteins. Moreover, they are mainly limited to a few model species. As a result, the ability to correctly predict protein function would accelerate research in fields related to any biological entity.

Monomers, the building blocks of proteins are called amino acids. An amino acid can be described as having a core carbon atom bonded to an amino group, a carboxyl group, a hydrogen atom, and an R group, which is the most important part of an amino acid and varies among all amino acids. A protein is essentially an amino acid chain in which each amino acid is linked to its neighbors by a peptide bond.

In the context of natural language processing (NLP), a protein can be thought of as a document while the amino acids that make up the protein up would be a sequence of characters with no clear boundaries between potential words/tokens. This kind of analogy raises questions such as "Do the properties of natural languages also apply to proteins?" or "Can a protein be converted into tokens by grouping some consecutive amino acids, with a token corresponding to something common and special?". Dozens of different questions like these could be asked, and one of the most interesting ones would be whether the functionality of a protein can be predicted from raw sequence data using state-of-the-art NLP models and/or traditional machine learning techniques.

Finding the functionalities of a protein based solely on its sequence data would be a major innovation in the field as the introduction of low-cost sequencing technologies

and the development of computational approaches have led to a tremendous increase in the number of sequences accessible in major protein sequence databases. One example of such a database is the UniProt Knowledge Base. At the time of writing, UniProtKB contains over 120 million sequences. This massive number of proteins is mainly discovered using computational methods, so only about 0.5% of these proteins are manually annotated. Because of the scale, design, and expense issues for tests to detect a function, it is reasonable to assume that most proteins with unknown functions will not be described experimentally. One of the most pressing biological problems in the era of modern biology is determining the functions and processes of newly identified proteins. While these improvements in sequencing are taking place in the biology domain, breakthroughs in experimental and computational biology techniques have also accelerated, but the results are not yet at a satisfactory level so the three-dimensional structures of many proteins are not available and will probably not be available soon. Moreover, the amino acid sequence itself should provide sufficient information to a certain point to determine the function from the sequence alone. Therefore, prediction of function from raw sequence data alone has been a hot topic over the past decade.

With this in mind, the main motivation for this thesis can be described as understanding whether using only the sequence information of the protein, which theoretically contains all the necessary information, can lead us to recognise the functionality of the protein. Two different NLP methods are adapted to the problem of protein function prediction based on the assumption of similarity between protein sequences and natural languages: (i) Bidirectional Transoformer BERT model (ii) Heterogeneous Graph Convolutional Network model. The results are compared to each other as well as with another study using the three-dimensional structural information in addition to amino acid sequences.

2. BACKGROUND

In this thesis, two different types of information about proteins have been used. The first one is the amino acid sequences of proteins which are treated as features, while the other one is the GO-terms, which represent the functional properties associated with these proteins, treated as the labels to predict.

2.1. Proteins

Proteins are molecular robots that are utilized inside or outside the cell to perform specific tasks. They serve a wide range of roles within cells, such as catalyzing/inhibiting metabolic processes, DNA/RNA replication, responding to a stimulus in the environment and transporting chemicals to a predefined destination. These tasks are determined by their three-dimensional structure as they bind to other molecules to alter the state of the cell.

2.2. Amino Acids

As mentioned in the introduction, proteins are composed of 20 unique amino acids. The three-dimensional structure and intracellular functions of each protein are dictated by the unique amino acid sequence and their arrangement. Folding is a spontaneous process controlled primarily by hydrophobic contacts, van der Waals forces, intramolecular hydrogen bond formation, and the resulting entropy. Therefore, ideally, all the information required to fold a straight chain of amino acids into a fully functional protein structure is hidden within the amino acid chain itself. The folding process is also affected by the type of solvent, salt ratio, pH, temperature, and the presence of cofactors and molecular chaperones, but the effects of these external factors are generally omitted because the intracellular environment has certain ranges for these external factors and these ranges are incredibly stable [1].

2.3. Protein Functionalities

As mentioned, proteins are the robotic macromolecules of living cells. Thus, each of them has one or more specific tasks to keep the cell alive and is transcribed to fulfill these tasks. These functionalities of proteins have been widely used as research topics by both molecular and computational biologists.

To have a standard in this context, all prescribed functionalities have been assigned a unique identifier called GO-term by the Gene Ontology (GO) consortium [2] and the Enzyme Commission (EC) [3]. In this study, only GO annotations are used instead of the EC annotations because GO covers broader categories of functionalities. The GO Consortium has also classified the functionalities into three main classes: Molecular Function, Biological Processes and Cellular Component. An example of GO-terms would be *GO:0060047* which represents the function of cardiac contraction categorized as BP. This thesis attempts to predict these functionalities from amino acid sequences since a conserved sequence generally implies a conserved function [4].

2.4. Databases

2.4.1. Protein Data Bank (PDB)

Protein Data Bank (PDB) is a database containing three-dimensional structures of proteins. It contains over 184,000 entries as of January 2022. A PDB entry is assigned a four-letter accession code and a one-letter chain code. For each of the proteins in the database, a PDB file is available in which the amino acid sequence of the protein can be found along with their coordinates. An example of PDB IDs would be *5XTH-S* which represents the cryo-EM structure of the human respiratory supercomplex. Protein or protein-ligand complex structures can be retrieved from PDB [5].

One of the information that a PDB entry lacks is functional information. To overcome this, another database, SIFTS, is used.

2.4.2. Structure Integration with Function, Taxonomy and Sequence (SIFTS)

Structure Integration with Function, Taxonomy, and Sequence (SIFTS) is a project in the PDBe-KB database that provides annotations from GO resources. As of January 2022, most of the entries in the PDB database, about 150,000, are functionally annotated in this database [6].

Since this thesis is concerned with determining the functional properties of proteins, the unannotated portion of the PDB database is omitted and only the annotated proteins are used.

2.4.3. UniProtKB/SWISS-Prot

UniProtKB/Swiss-Prot is UniProtKB's professionally curated component produced by the UniProt consortium. It provides descriptions of hundreds of thousands of proteins, including function, domain structure, subcellular localization, post-translational modifications, and functionally described variations [7].

As mentioned in the introduction, SWISS-Prot contains over 120 million sequences as of January 2022. Since the vast majority of proteins are found via computational approaches, only about 0.5% of these proteins have been manually annotated. In addition to this massive number of missing functional annotations, the entries in this database also lack the three-dimensional information of the amino acids. In this thesis, the proposed model does not use the three-dimensional information of the proteins, so this is not a problem. Since it already contains the functional properties (e.g., GO annotations) of proteins, this database was used directly.

2.5. Technical Background

There is some technical background information that needs to be known to understand the approaches and methods used in this thesis.

2.5.1. Transformers

The architectural concept of Transformers was originally developed to convert a sequence input into a different sequence output. This includes any task that converts an input sequence to an output sequence, including translation, speech recognition, text-to-speech conversion, etc.

In this architecture, a stack of Encoder and Decoder layers is used. To avoid ambiguity, we refer to an individual layer as an Encoder or a Decoder and a combination of Encoder layers as an Encoder stack or a Decoder stack. Each Encoder and Decoder stack has its own Embedding layers for its respective inputs. The Encoder contains the crucial self-attention layer, which computes the connection between words in the sequence, and a Feed-forward layer. The self-attention layer allows the model to examine the input sequence while encoding a particular token. In the Decoder, there is a self-attention layer, a feed-forward layer, and a second Encoder-Decoder attention layer present. Each of the Encoder and Decoder layers has their own set of weights [8].

The Transformer's use of Attention is critical to its ground-breaking performance. When the model is processing a word, Attention allows it to focus on the other words in the input closely related to that term. For instance, there is a close relation between the words *building*, *small* and *going* whereas there is no real relationship between the words *small* and *student*.

The student is going to the small building.

By linking each word in the input sequence to every other word, the Trans-

former architecture uses self-attention to guide the processing of a sequence of words. By maintaining these word-to-word relationships, the architecture successfully learns the structure of that particular language. Any sentence in that language would be processed with a state-of-the-art accuracy for later tasks such as translation, text classification, etc. One of the most successful language models (LM) developed based on this concept, BERT, was chosen for the present work.

2.5.2. Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is an article published by Google AI Language researchers [9]. It has generated interest in the Machine Learning community as it shows cutting-edge results on a wide range of NLP tasks, such as Question Answering (SQuAD v1.1), Natural Language Inference (NLI), and others.

The use of Transformer's bidirectional training for language modeling is the fundamental technological breakthrough of BERT. Previously, researchers have dealt with a text sequence by looking only at the left-to-right perspective. The research findings suggest that a bidirectionally trained language model can have a better understanding of the language context than a unidirectional language model. In the study, the researchers describe a unique approach called Masked LM (MLM) that enables bidirectional training of such models that were previously unattainable [9].

Transformer, as explained above, is the primary building block of BERT. Since the purpose of BERT is to construct a language model, only the Encoder mechanism is needed. Therefore, only the relevant part of the Transformer mechanism is used directly in the architecture [9].

To train the model, 15% of each sequence is replaced with a *[MASK]* token before feeding word sequences into BERT. Each document is tagged from the beginning with another keyword called *[CLS]*. Based on the context provided by the unmasked

words in the sequence, the model then attempts to estimate the original value of the masked words. The output word prediction is achieved by adding a classification layer on the output of the Encoder, multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension, and finally computing the probability of each word in the vocabulary using Softmax. This is how a vanilla BERT model is trained.

After a training process, the model is supposed to learn the inner dynamics of this specific language so that a downstream task (e.g., classifying texts with one or more labels) can be done by adding an appropriate architecture to the pretrained language model. For instance, a classification task such as sentiment analysis can be achieved by simply adding a new classification layer on top of the last Transformer output of the *[CLS]* token representing the entire document in a single vector.

In this thesis, it was assumed that this approach could be used to predict the protein functionalities from amino acid sequences by treating each sequence as a document. However, since BERT is an enormously large architecture with 345 million parameters, training it from scratch would take too long and require high-performance hardware (e.g., a high-tier GPU) which we did not have at the time. A little research on the pretrained models revealed that multiple research groups had already done some prior work [10,11], and one of them [10] was chosen to train further for our downstream task of protein function prediction.

One of the disadvantages of using a pretrained model is that the input scheme cannot be changed afterward. In our case, unlike the tokenization method explained below, this limitation forced us to treat each amino acid as a word itself since training from scratch with the tokenized version was not an option.

2.5.3. Byte-Pair Encoding (BPE)

Proteins and other chemical compounds can be represented as strings of characters. We use the primary structure, the amino acid sequence, to represent the proteins in this work. An example amino acid sequence might look like this

MKTAYIAAKQRQISFVKSHFSRQLEERLAAG

where each of the characters stands for an amino acid. A sequence can have at most 20 unique characters because there are 20 unique amino acids available. Therefore, the following analogy could be made:

- An amino acid is a letter, and the alphabet has 20 unique letters
- An amino acid sequence is a sentence composed of these letters
- A group of amino acids is a word, but the boundaries of these words are removed so that the words are written side-by-side without spaces between them.

This type of interpretation also explains the general behavior of proteins because a protein is a robot made up of different parts to accomplish its task, such as localizing itself in intracellular space, binding a molecule by chemically recognizing it, and processing the bound molecule. They also have different parts to become inhibited or catalyzed in different cellular conditions to change their action by the other molecules in the cell [12, 13].

This analogy opens us the possibility to apply natural language processing approaches to protein sequences. However, most of these techniques are based on the words/tokens, which are not apparent here due to the non-spaced nature of proteins. A naive way of processing the documents is to consider each amino acid as a word rather than a letter so that the above example becomes the following:

M K T A Y I A A K Q R Q I S F V K S H F S R Q L E E R L A A G.

The problem here is that the definition of a word is squeezed into the individual letters so there are only 20 unique word possibilities to explain thousands of different functionalities. There must be a group of amino acids that form a larger word that hopefully corresponds to a functionality alone or as a combination with other amino acid groups. It is also important to detect the possible amino acid groups found in different proteins, as this is a result of evolutionary processes to find the relationships between sequences and functions.

In this context, byte-pair encoding (BPE) was chosen to identify the tokens in a protein corpus. BPE is a basic data compression method in which the most frequent pair of consecutive bytes of data is replaced by a byte that does not occur in that data [14]. It has been used in the literature for many protein and biochemistry tasks [15–17]. Compared to other tokenization methods, this technique gave comparable or superior results [15].

Suppose we have *aaabdaaabac* that needs to be encoded. Since the byte combination *aa* is the most frequent combination, we will replace it with Q, since Q does not occur in our data. As a result, we now have *QabdQabac*, where Q=*aa*. Replacing the next common byte pair, *ab*, with W results in $QWdQWac$, where Q=*aa* and W=*ab*. The only remaining byte pair is *ac*, which appears as a single byte, so we do not encode it. We can encode QW as E using recursive byte pair encoding. Our data, *aaabdaaabac*, has now been translated into *EdEac*, where E=QW, W=*ab*, and Q=*aa*. It cannot be encoded further since no byte pair occurs more than once. The original data can be decompressed by replacing the encoded values in reverse order.

BPE guarantees that the most common terms in the vocabulary are represented as a single token, while uncommon words are split into two or more subword tokens. In addition, the shortening of sequences minimizes computational costs. This is particularly important for proteins with extensive amino acid sequences.

The given example works on a single sample protein sequence, but the datasets

used in this thesis, especially the SWISS dataset, contain more than 250,000 different proteins. The pairs are computed over all these proteins by considering all proteins as one large document. Such vertical byte pair inspection allows finding more conserved pairs and hopefully discovering the best function-word relations. As a result, the amino acid sequence given above would be tokenized as follows, rather than the naive letter-wise splitting:

MKT AYIAAK QR Q ISFVKSH F S RQLEER LAA G.

2.5.4. Graph Convolutional Networks (GCN)

In graph theory, a graph is a structure consisting of a set of objects, some of which are connected. The main entities of a graph are the vertices, also known as nodes, and the edges that connect two vertices, either directionally or undirectionally, depending on the type of the graph [18].

This is a concept relevant to this thesis because a protein, especially in its three-dimensional form, can be directly transformed into a graph by treating the amino acids as nodes and the hypothetical links between the spatially close amino acids as edges. Such an interpretation has been a hot topic in computational biology since 2015 when some ground-breaking approaches were started to be defined [19–22].

It is challenging to generalize neural models such as RNNs or CNNs to work on arbitrarily structured graphs. Recent studies have generally focused on bridging the gap between fast heuristics and the slower but more principled spectral methods. [23] uses Chebyshev polynomials with free parameters learned in a neural network-like model to approximate smooth filters in the spectral domain. They produce convincing results in regular-domain tasks (such as MNIST) comparable to those of a basic two-dimensional CNN model.

Since then, better approaches have been studied. In [24], for example, they take

a similar approach and begin with the framework of spectral graph convolutions, but introduce some simplifications and new suggestions to earlier work which in many cases allow for significantly faster training times and higher predictive accuracy. It also yields state-of-the-art classification results on several benchmark graph datasets. After a few years, at the time of writing this thesis, this approach is still accepted as the most straightforward and mathematically best approach for GCN architecture among all the attempts in the literature and cited more than twelve thousand times [25].

Most GCN models have a relatively uniform design. These models are called convolutional models because the filter parameters are often shared at all places in the graph. The goal of a GCN model is to learn a function of signals/features on a graph $G = VE$ where G represents the graph, V represents the vertices, and E represents the nodes.

There are two inputs to a GCN layer. The first one is an $N \times D$ feature matrix X, while the second one is an $N \times N$ adjacency matrix A, which is a representative description of the graph structure in matrix form. Here, N represents the number of nodes, and D represents the number of input features. The layer generates a node-level output Z, which is an NxF feature matrix, where F is the number of resulting features per node. Since a neural layer can be expressed as a nonlinear function, the mathematical representation of what happens in any GCN layer can be written as follows:

$$H^{(l+1)} = f(H^{(l)}, A), \tag{2.1}$$

with $H^{(0)} = X$ and $H^{(L)} = Z$ where L is the number of layers. When this function $f(.)$ is rewritten in detailed form, it is

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}), \tag{2.2}$$

where $W^{(l)}$ is the weight of layer l and $\sigma(.)$ is a non-linear function such as ReLU.

This is the general mathematical formula of how a neural network passes the input from beginning to end, except the adjacency matrix integration. The main difference that the GCN brings is the addition of this matrix into the calculations.

The use of the adjacency matrix is tricky and, in the naive form, requires additional heavy matrix multiplications, including iteratively summing the feature vectors of all nearby nodes for each node. The method proposed in the aforementioned paper [24] remedies this by adding an identity matrix I to A to force the addition of hypothetical self-loops of nodes. Another optimization done by [24] solves the problem of having an unnormalized adjacency matrix leading to major normalization issues as the layers propagate due to multiplications. What they suggest is normalizing A such that all rows sum to one by using $A^{'} = D^{-1}A$ where D is the diagonal node degree matrix. They further optimized this by multiplying by the inverse square root of D from both sides of A like $A^{'} = D^{-1/2}AD^{-1/2}$. Therefore, the final version of a layer output calculation becomes

$$f(H^{(l)}, A) = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l)}W^{(l)}) \tag{2.3}$$

$$\hat{A} = A + I, \tag{2.4}$$

where $\hat{D}$ is the diagonal degree matrix of $\hat{A}$.

The output of GCN explained so far does not correspond to what this thesis offers. In this standard form, GCN outputs an embedding for each node. However, the primary goal of this thesis is to classify proteins into potential intracellular functionalities based on their amino acids. In the *Methods* sections, it is explained how this approach is transformed to support this objective.

2.5.5. Point-wise Mutual Information (PMI)

In NLP, Point-wise Mutual Information (PMI) is used to mathematically represent the association of word pairs forming a unique concept when the words come together.

This can be understood well by an example. Suppose there are two words, *social* and *media*. By themselves, they have different meanings, but when they are used together as *social media*, the overall meaning changes drastically and forms a unique concept. The meaning does not have to change all the time. Rather, it may be reinforced, negated, or emphasized by the combination. However, as in the example, finding the most useful concept formations is an important aspect for general NLP tasks, so is for our study.

The point-wise mutual information approach can mathematically evaluate this type of co-occurrences. It is calculated by the likelihood of two words co-occurring with the penalty for occurrences of the terms without the other one. Mathematically, the method calculates the (logarithmic) probability of co-occurrence divided by the product of the marginal probabilities as shown below:

$$PMI(a,b) = log\left(\frac{P(a,b)}{P(a)P(b)}\right), \tag{2.5}$$

where a and b are the words, $P(a,b)$ is the probability of co-occurrence (i.e., the joint probability) of these words, $P(a)$ and $P(b)$ are the individual probabilities of occurrence (i.e., the marginal probabilities) of these words. If the co-occurrence of a pair of words is high, but their marginal probabilities are low, it means that they usually co-occur and therefore have a high PMI value and vice versa.

In this thesis, PMI was used to calculate the weights between two word nodes for the GCN model.

2.5.6. Term Frequency-Inverse Document Frequency (TF-IDF)

The importance of a term to a document can be measured by TF-IDF. Term frequency refers to the number of occurrences of a term in a document. In our case, it corresponds to the number of occurrences of an amino acid group in a protein. The mathematical representation of the term frequency is

$$TF(w,d) = \frac{\text{\# occurrences of w in d}}{\text{Total \# words in d}}, \tag{2.6}$$

where w is the subject word and d is the subject document.

Inverse document frequency refers to the inverse ratio of the number of documents in which the word occurs. In our case, it corresponds to the inverse ratio of the number of proteins in which an amino acid group occurs. It's mathematically formulated as

$$IDF(w,C) = ln\left(\frac{\text{Total \# documents in C}}{\text{\# documents containing w}}\right), \tag{2.7}$$

where w is the subject word and C is the whole corpus.

TF-IDF is the product of these two separate measurements. If a term occurs more frequently in one document than in all other documents, TF-IDF gives a significant score, which means there is strong evidence that the term is important for that particular document.

In this thesis, TF-IDF is used to calculate the weights of the edges between a word node and a document node.

2.6. Related Work

In the literature, various methods have been applied to sequence data to predict three-dimensional structure or functions of proteins. The existing methods can be

divided into three main categories based on the type of approach studied: (i) statistical approaches, (ii) traditional machine learning-based approaches, and (iii) Deep Learning approaches.

Probably the oldest and yet incredibly successful model, long considered the gold standard in this context, is BLAST [26]. It looks for similarities between biological sequences by comparing nucleotide or protein sequences to sequence databases and calculating statistical significance. As sequencing technologies continue to evolve at an accelerating pace, this database and associated software continue to improve, and it is still the baseline for the functional assessment challenges such as Critical Assessment of Functional Annotation (CAFA) [27]. However, the quality of manually created features, either from sequence or structure, often significantly impacts on the success of these approaches because the features rely heavily on heuristics which in turn require expertise, and in some cases unstable assumptions, thresholds and preprocessing pipelines.

Scientists have used various machine learning algorithms with hand-crafted features derived from the chemical information of amino acids and their pair-wise relationships [28, 29].

One of these studies uses Hidden Markov Models [28] and their proposed approach does not perform better than the traditional methods such as BLAST. This is because these approaches are relatively old so available information at those times is not comparable to the last few years.

A relatively important old school study is FFPred [29], which uses feature engineering-based machine learning methods. It is a gem from this era, showing that different sequence features are informative for different functions.

Another important model study is FunFams [30]. In this study, they compute a functional coherence index based on conserved sites in sequence alignments to find an

appropriate cut for a hierarchical clustering tree of sequence relations. It is shown that the proposed approach provides outstanding results compared to any older model.

Up to this point, more traditional approaches have been studied, but since the early 2010s, the perspective has changed. The number of study groups working on this topic has been increasing ever since as the protein databases grow faster than ever. At the same time, the computational power brought by the GPU industry allows us to build larger neural models both horizontally and vertically. These paradigm shifts have led computational biologists and scientists to look more deeply into this topic. Still, they have generally sought to predict the three-dimensional structure of proteins rather than the functionality itself. These are related problems because correctly predicting the three-dimensional shape of the protein would inadvertently provide functional information. However, most of the work in the literature does not report such information and primarily refers to spatial information in three-dimensional space [31].

The first Deep Learning method used for this goal was Multi-Layer Perceptron [32, 33]. The reported results are somewhat comparable to the results of BLAST, but were easily outperformed by more complex models like convolutional and recurrent neural networks.

Deep Learning algorithms have made tremendous performance gains over the past decade on various topics, from learning protein sequence embeddings for contact map prediction to predicting protein structure and function. In particular, CNNs have demonstrated significant performance in solving computational biology challenges [34–36].

CNNs have enabled task-specific features to be extracted directly from a protein sequence or its three-dimensional structure, circumventing the limitations of feature-based ML approaches. The vast majority of sequence-based protein function prediction algorithms use one-dimensional CNNs or variants of them searching for recurrent spa-

tial patterns within a given sequence and transforming them hierarchically into complex features using multiple convolutional layers [34–37].

The best-studied approach is DeepGO [37], one of the state-of-the-art deep learning methods, where a neural embedding layer was used to embed all possible amino acid triplets into a 128-dimensional space, and a convolutional neural network is applied to these triplet embeddings. The reported results were state-of-the-art when the paper was written in 2018.

Some studies have recently used three-dimensional CNNs to make predictions and extract features from protein structure data [38, 39]. These approaches use a three-dimensional volumetric protein structure given on a grid as input. Storing explicit three-dimensional representations of protein structure at high resolution is not memory efficient. Moreover, in most cases, three-dimensional CNN would convolve over the empty space, which is relatively wasteful. Geometric deep learning approaches, GCNs, have recently provided a solution to circumvent these restrictions by generalizing convolution processes on more natural graph-like molecular representations [24, 40]. GCNs have demonstrated significant performance improvement on a variety of challenges ranging from learning meaningful molecular fingerprints for predicting the biochemical activity of drugs to predicting protein interfaces [19, 20, 41].

One of the most recent studies [42] compares several different Deep Learning approaches, including MLP, CNN, and GCN. It also compares the results with unsupervised learning methods and physics-based methods. It's reported that using Deep Learning approaches produces superior results compared to the other methods. One of the interesting results presented here is that using shared parameters for different amino acids in the sequence extends their models' ability. That means the order of the amino acids and preserving their sequential information is essential to improve performance rather than using the sequence as a single unit.

The most successful work in this context, DeepFRI [25], also uses GCN, but it uses three-dimensional information in addition to the sequence information. The sequence is given to an LSTM model, jointly trained with the GCN model, to convert the sequence into an embedding. Its success is understandable, as GCNs are expected to achieve better results when the actual graph-like structure is known and available. The use of more information about proteins increases the prediction accuracy in this work. Furthermore, they have also reported that using an additional model, LSTM, to learn how to embed proteins increases their accuracy. The drawback of this approach is that the three-dimensional information of proteins is mostly unavailable in 2022. The best way to fill in this missing information is to use a structural prediction algorithm before using this type of approaches. However, this is not ideal because errors propagate throughout the process since the information is synthetic. Since this research achieves the best results in the literature, it is the main competitor in this thesis.

3. MATERIALS AND METHODS

3.1. Data

As explained in the *Databases* section, two different databases were used in this work. One is PDB, and the other is SWISS-Prot. The datasets used in this work were mainly chosen by [25]. They curated these datasets from the mentioned databases by applying some specific filters to the sequences. The most important notes about the datasets can be found below:

- PDB25 database has been chosen. PDB25 defines the dataset containing only the proteins with maximum identity to the other proteins in the dataset at most 25%. A similar filtering is done to the sequences in SWISS-Prot database.
- The test and validation sets are randomly sampled from these pre-filtered datasets by considering the sequence similarity. If a sequence selected for the test or validation set is similar to training set more than 20%, it is replaced with a new one.

The datasets provided by the authors of [25] contain only the sequence information, as they decided not to publish the functional mapping. Therefore, the functional annotations were done manually by retrieving the GO-terms from the SIFTS database dump with the specific versions mentioned in the paper (i.e., PDB release 08.19 and UniPort release 2019.02). There are three different types of functional class types: Molecular Function (MF), Biological Process (BP), and Cellular Component (CC). According to the authors, they also filtered the GO-terms based on the following criteria:

- Removing GO-terms without the following evidence codes: EXP (inferred from experiment), IDA (inferred from direct assay), IPI (inferred from physical interaction), IMP (inferred from mutant phenotype), IGI (inferred from genetic interac-

tion), IEP (inferred from expression pattern), TAS (traceable author statement) and IC (inferred by curator)

- Removing the GO-terms having not enough training examples. The lower limits for the class types are:
  - Molecular Function (MF) and Cellular Component (CC) - 30 unique sequences
  - Biological Processes (BP) - 50 unique sequences

After applying these filters to the GO-terms and matching them to the data, the datasets were ready for experimentation. In Table 3.1, some statistics about the datasets can be found.

Table 3.1. Summary statistics of datasets.

Class Type	**# Classes**	**Dataset**	**# Proteins**	**# Training**	**# Valid**	**# Test**
MF	489	SWISS	228,817	202,807	22,595	3,415
		PDB	31,125	24,961	2,749	
		SWISS + PDB	256,527	227,768	25,344	
BP	1,943	SWISS	232,452	206,123	22,914	
		PDB	29,562	23,522	2,625	
		SWISS + PDB	258,599	229,645	25,539	
CC	320	SWISS	119,838	104,772	11,651	
		PDB	16,018	11,304	1,299	
		SWISS + PDB	132,441	116,076	12,950	

The minimum number of terms among these class types belongs to CC and is 320, while the maximum number of classes belongs to BP and is 1943. A given protein can have zero or more intracellular functions for each of these class types. This means that the problem we are dealing with is fundamentally a multi-label classification task. Each class type is trained with three different dataset combinations: (i) PDB only, (ii) SWISS-Prot only, and (iii) the merge of these two datasets. The test set is the same for all the experiments and derived from PDB dataset.

One of the main problems of a multi-label classification task is that there is generally not enough data for some of the labels. One can think of the labels as a matrix with 1s and 0s. In our case, most of the cells in this matrix consist of 0s. The sparsity level of any of these combinations is at least 99.98%, and this makes the problem extremely difficult to obtain satisfactory results. Figure 3.1 shows the histogram of samples in terms of the number of functions associated with them for MF class type in the PDB dataset. It should be noted that there are 489 unique functions defined in MF, but the samples associated with the highest number of functions have 43, 36, and 36 functions in test, train and validation sets, respectively. The number of samples is getting higher as the number of associated functions is getting lower. The same analysis is done for the other class types and can be seen in Figures A.1 and A.2.

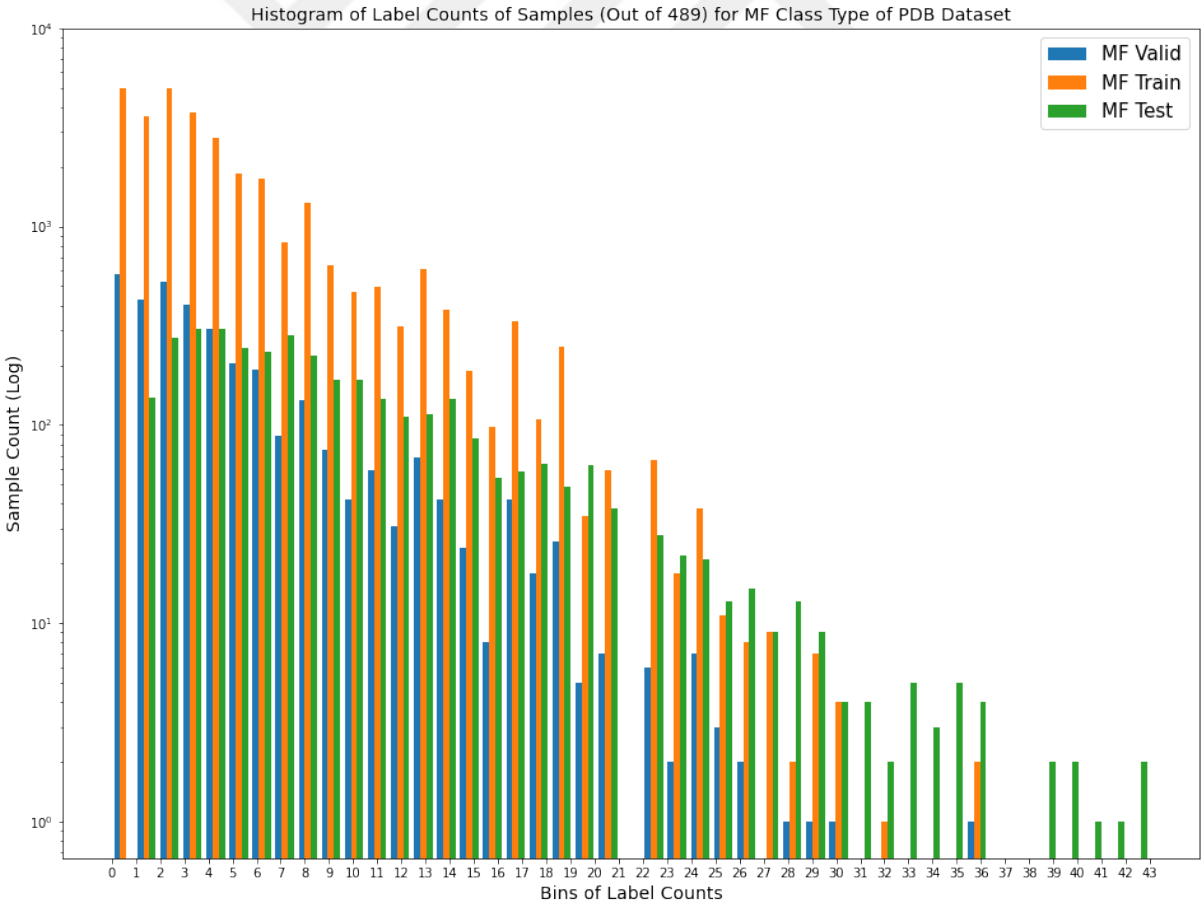

Figure 3.1. Histogram of samples in terms of number of functions associated with them for MF class type in PDB dataset.

This is the way of visualizing the extreme sparsity in terms of the samples. A similar analysis can be done from the perspective of classes as well. In Figure 3.2, the sparsity of the datasets is shown by a cumulative histogram where x-axis corresponds to the positive label counts of the samples and y-axis corresponds to the cumulative likelihood. To demonstrate how these histograms should be read, an example would be given as the following: In the top plot, around 218 samples, corresponding to $\frac{218}{31125} = 0.7\%$ as a ratio to the total samples, have the portion of 75% of all positive labels. This means that the remaining 99.3% of the samples have only 25% of the positive labels, indicating that the number of useful (i.e., positively labeled) samples are too low for most of the functions.

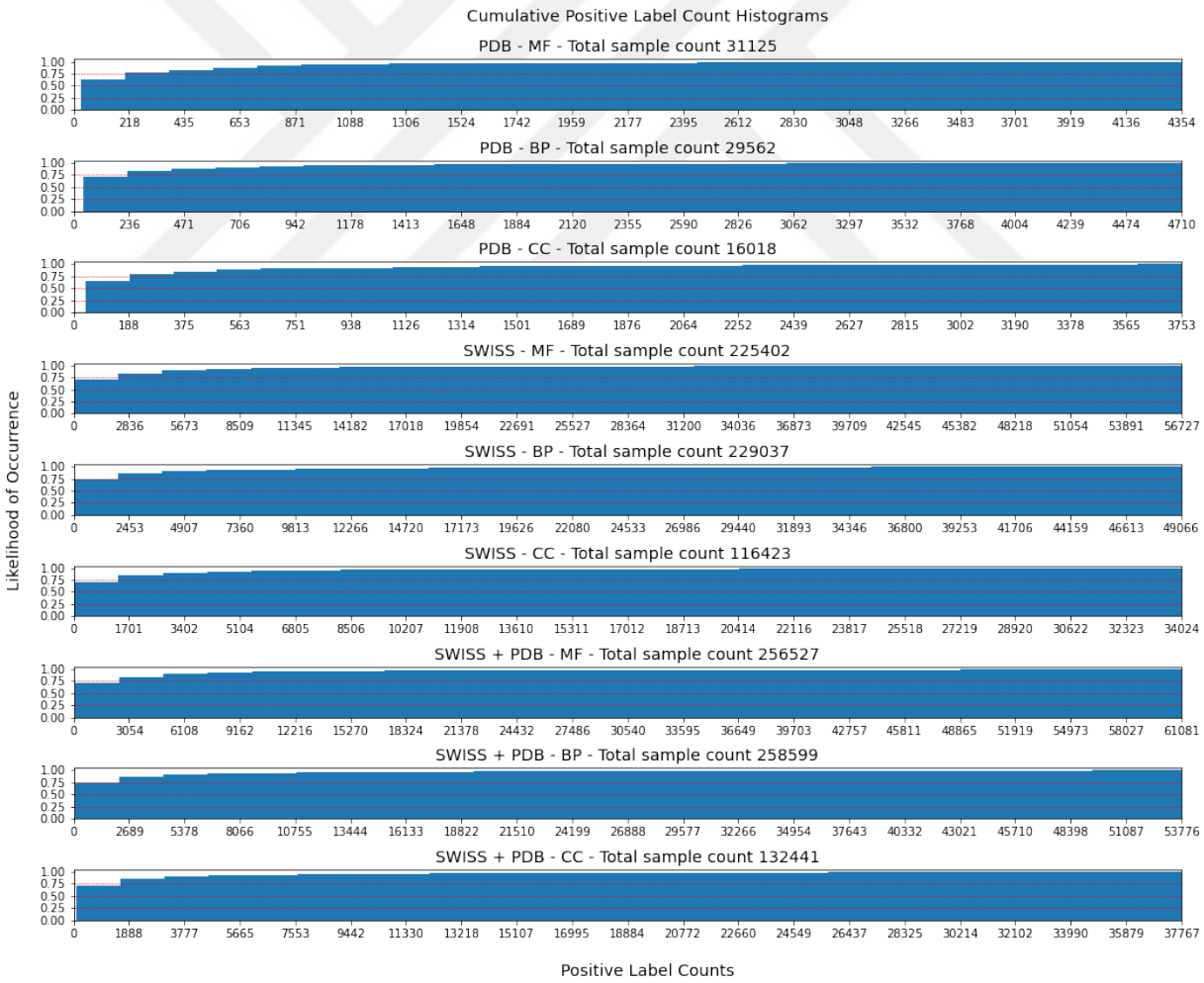

Figure 3.2. Cumulative positive label counts indicating the sparsity of the labels for each dataset.

When Figure 3.2 is inspected, for all the cases, at least 75% of the functions are associated with at most 0.1% samples. This means that around 99.9% of the entire sample set is not associated with this 75% portion of the label set. Further analysis reveals that at least 95% of the functions are associated with at most 3.33% of the entire samples.

Figures A.4 and A.3 show the same data zoomed in to the functions associated with the 100 and 1000 samples respectively. When inspected, some of these highly negatively labelled classes have even lower ratio of positively labelled data. This gives an insight about the potential performance of the models even before the training/testing processes. It is expected that the performance of the models for these functions will be quite low.

At this point, the data and the objective are converted into the following concepts to formalize the problem from NLP perspective and process the data:

- A word (or a token) is defined as the smallest linguistic unit and refers to an amino acid group whose length is x s.t. $x \in \mathbb{Z}^+$.
- A document is defined as a group of words and refers to a protein sequence whose length is l s.t. $l \in \mathbb{Z}^+$.
- A label (or a class) associated with a document refers to a function associated with a protein.

All these datasets are subjected to BPE tokenization because, as explained in the *BPE* section, an amino acid group and its association with other groups may fit a functionality better than the individual amino acid associations. Moreover, in [4], it is defended that a conserved sequence implies a conserved function, and a conserved sequence can correspond to a word in our concept. In Tables 3.2 and 3.3 the resulting statistics after applying BPE tokenization with increasing tokenization steps can be seen. Here, a *step* corresponds to finding a new amino acid group pair and merging them into a new group. There is no merged group at the beginning, so all the groups

consist of only one amino acid initially. As the tokenization continues running, longer and more complex amino acid groups start to emerge step by step.

Table 3.2. Summary document statistics for different BPE tokenization settings.

Dataset	**BPE Step**	**Avg Doc Length ± Std**	**Max Doc Length**
PDB	0	282.12 ± 170.4	1000
	20,000	140.22 ± 205.69	492
	50,000	97.81 ± 228.89	388
	70,000	95.24 ± 227.35	346
	100,000	92.09 ± 225.50	340
SWISS	0	277.96 ± 162.95	1000
	20,000	136.42 ± 198.34	516
	50,000	93.16 ± 215.88	365
	70,000	87.13 ± 213.11	351
	100,000	83.65 ± 210.96	346
SWISS + PDB	0	279.12 ± 164.68	1000
	20,000	137.54 ± 200.01	516
	50,000	94.67 ± 219.56	388
	70,000	89.24 ± 217.42	351
	100,000	84.96 ± 214.97	346

Table 3.3. Summary word statistics for different BPE tokenization settings.

Dataset	**BPE Step**	**# Unique Words**	**Avg Word Length ± Std**	**Max Word Length**
PDB	0	20	1.00 ± 0.00	1
	20,000	19,856	1.44 ± 0.91	13
	50,000	49,652	2.88 ± 1.31	46
	70,000	69,326	2.96 ± 1.38	50
	100,000	99,153	3.06 ± 1.47	68
SWISS	0	20	1.00 ± 0.00	1
	20,000	19,901	2.25 ± 1.05	22
	50,000	49,739	3.01 ± 1.15	48
	70,000	69,402	3.19 ± 1.28	54
	100,000	99,189	3.33 ± 1.45	68
SWISS + PDB	0	20	1.00 ± 0.00	1
	20,000	19,920	1.98 ± 1.01	22
	50,000	49,873	2.98 ± 1.1	48
	70,000	69,474	3.04 ± 1.22	54
	100,000	99,219	3.21 ± 1.46	68

The word length distribution of some selected combinations can be seen in Figure 3.3. These plots show that BPE tokenization has been successfully applied to the datasets, and it can be said that the resulting tokens are commonly available in the datasets based on their frequency. This fits the conserved subsequence assumption explained in the *Protein Functionalities* section, which would increase the prediction performance. To measure the potential performance improvement, all experiments were reproduced for the non-tokenized version as well. The experimental results are reported in *Results* section.

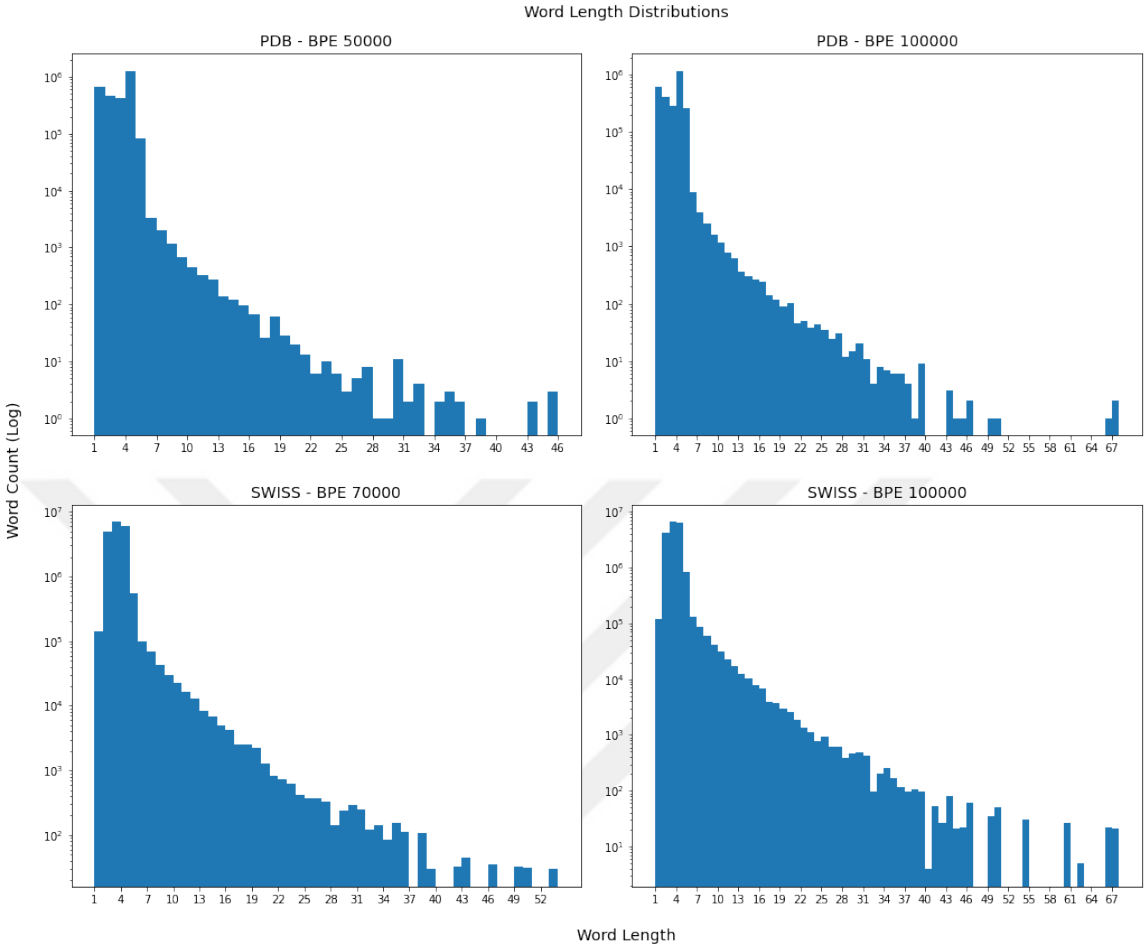

Figure 3.3. Word length distribution of encoded datasets for different BPE tokenization settings.

3.2. Models

There are two state-of-the-art models used in this thesis. One is a BERT model variant, while the other is a GCN model variant.

3.2.1. Prot-BERT and Fine-Tuning

As described in the *Methods* section, one of the models used in this thesis is a BERT variant called Prot-BERT developed by the authors of the paper [10]. The main motivation for using this model is both that it is one of the best performing models in the NLP literature and that it implicitly treats protein and its atomic building blocks

as linguistic entities (i.e., sentence and words, respectively) by preserving the sequential information leading to the potential functional mapping.

Prot-BERT is a pretrained BERT-large model. This model and other variants (e.g., ProtXLNet, ProtAlBERT, etc.) are published to Huggingface NLP model library by the study group, Rostlab [43]. The repository of the Prot-BERT model [44] contains the model's PyTorch binary file, a vocabulary text file containing all the words (i.e., 20 amino acids as explained in the *Models* section), and a configuration file containing information about the hyperparameters of the models.

Since the binary of the model is given in PyTorch format, PyTorchLightning framework, a lightweight wrapper for PyTorch machine learning framework [45], is used to fine-tune the model for the main objective of this thesis. As mentioned earlier, the model splits the input sequence into individual amino acids, so it does not support any grouped format. To make the model support such tokens, it must be trained with a suitable vocabulary and corpus from scratch. However, training it from scratch was not an option because the underlying model, BERT, is a massive architecture with more than 345 million parameters, and the hardware was limited.

The only option was to fine-tune the pretrained model, and the BERT version was preferred over the AlBERT and XLNet versions because BERT is their primary architecture. While the other versions have been developed to make the training faster and more optimized, sticking with the original version would make the results more comparable to the models in the literature. It is also the version mainly evaluated in its article [10].

The fine-tuning process starts from a pretrained model that had already been trained with a tremendously large dataset to accomplish Masked Language Modeling objective in a self-supervised fashion. In the case of Prot-BERT, this dataset is *Uniref100* which consists of 217 million different protein sequences [10]. In a train-from-scratch approach, all weights in the model are randomly initialized and are close

to zero, whereas using a pretrained model allows us to start from model weights that are working at least for some specific tasks and can be further optimized for other downstream tasks.

The model is capable of embedding proteins into 768-dimensional vectors as this was the native output of the model. This feature suggests that embeddings can capture important biophysical aspects that determine the shape of the proteins so that their functionalities [10].

Thus, as the first step of the fine-tuning process, a classification head is attached to the model that takes the embedding of the protein from the output of *[CLS]* and feeds this embedding to this newly introduced classification head. The number of output nodes equals the number of functions in each class type. This head may have one or more layers as it is a fully connected structure on its own.

3.2.2. Text-GCN

In the *Models* section, vanilla GCN was explained. Convolution is useful for our objective because the amino acid sequence captures the functional information by some local groups, not by individual presences of amino acids. The vanilla version of GCN can capture high-order neighborhood information and solves the problem of having a vast sparse adjacency matrix for convolution purposes by convolving only over the connected nodes (i.e., presence of an edge between the nodes) rather than convolving over the entire matrix.

In this vanilla version, the nodes correspond to entities related to each other. However there are two types of entities in our case: (i) proteins as documents and (ii) amino acid groups forming these proteins as tokens. Therefore, both entities must be represented somehow in the graph. The solution comes from a study [46] in which the network is heterogeneous and contains the word nodes and the document nodes simultaneously. In this version, called Text-GCN, the number of nodes, $|V|$ is equal

to the sum of the number of documents (i.e., proteins) and the number of the unique tokens (i.e., amino acid groups). The following rules are applied to connect the nodes with edges:

- There is an undirected edge between a word node and a document node if the word occurs in the document at least once
- There is an undirected edge between two word nodes if these words co-occur in the corpus
- There is no edge between document nodes

The weights of these edges (i.e., the adjacency matrix passed to the model) are calculated based on different calculations for different edge types. For edges between word nodes, the weight is calculated via point-wise mutual information (PMI), a measure of association information of word pairs, while the weight of an edge between a document and a word node is calculated via term frequency-inverse document frequency (TF-IDF) of that word in that particular document. PMI values are taken if they are positive (i.e., high semantic correlation), while a negative value means that the pair contains little or no information. The summary of weight assignments is shown below:

$$A_{i,j} = \begin{cases} PMI(i,j) & i \text{ , } j \text{ are words and } PMI(i,j) > 0 \\ TFIDF(i,j) & i \text{ is word, } j \text{ is document} \\ 1 & i = j \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

The probabilities of word co-occurrences are calculated as explained in the *Technical Background* section, but for our purpose, the co-occurrence does not mean that they are adjacent. This is because the functionality of a protein is hidden in its three-dimensional shape, and spatially close words lead to a specific function together. The words that are distant in the one-dimensional form may be spatially close to each

other in the three-dimensional folded form. Therefore, two words are considered to occur together if they occur in a fix-sized sliding window. Thus, calculation of PMI becomes:

$$PMI(i,j) = log\left(\frac{P(i,j)}{P(i)P(i)}\right) \tag{3.2}$$

$$P(i,j) = \frac{\#W(i,j)}{\#W} \tag{3.3}$$

$$P(i) = \frac{\#W(i)}{\#W}, \tag{3.4}$$

where i and j are the words, $\#W(i)$ is the number of sliding windows the word i occurs, $\#W(i,j)$ is the number of sliding windows the words i and j co-occurring and $\#W$ is the total number of sliding windows in the corpus.

The number of GCN layers is also an important concept here. Two-layer GCN allows message transfer between nodes that are no more than two steps apart. Although there are no direct edges between document nodes in the graph, two-layer GCN permits information to flow between documents pairs indirectly. More layers would bring complex relationships.

3.2.3. Loss Functions

Loss functions are the objective functions that mathematically define how a Deep Learning model learns the task of interest by minimizing the output of this function as much as possible.

The main goal of this thesis is a the multi-label classification task in the machine learning literature due to the fact that a protein can have multiple functionalities rather than a single one. For such objectives, each output must be evaluated separately, which means that the loss (i.e., forward propagation cost) in the training process must be calculated by considering all labels separately.

For a regular multi-class classification task where an input has a single associated label, one commonly used loss function is the Softmax Loss Function. However, it does not work here because the model learns to output only the most suitable option with Softmax instead of multiple (i.e., zero or more) options. To overcome this problem, each output should be treated as a separate possibility that can be achieved using the Sigmoid function on each output. Then, the output of the Sigmoid function is thresholded for each class to binarize the results. Learning each class separately led the study to use two possible binary loss functions, Binary Cross Entropy Loss and Focal Loss.

3.2.3.1. Binary Cross Entropy Loss. The idea of cross entropy has its roots in the field of information theory, where Claude Shannon established the notion of entropy in 1948 [47].

Binary Cross Entropy Loss compares each predicted probability (i.e., the outputs of the Sigmoid activation function) which are real numbers between 0 and 1, with the actual class labels, which is 0 or 1. The score is then calculated by penalizing the probabilities depending on their distance from the true labels. That is, how close or far the predicted value is from the actual label.

The vanilla version of Binary Cross Entropy Loss is:

$$H(y, \hat{y}) = -\frac{1}{N}\sum_{j=1}^{N}[y_j log\hat{y}_j + (1 - y_j)\, log\,(1 - \hat{y}_j)], \tag{3.5}$$

where y is the true labels, $\hat{y}$ is the prediction results (i.e., output of the Sigmoid), and N is the number of samples. This is one of the loss functions tried in this thesis.

In our case, the label matrix is highly sparse (99.98% on average) such that the positive samples are extremely rare compared to the negative samples. Such an imbalanced dataset in terms of the label polarity would introduce the problem of having

a model predicting an output as negative and accurate almost all the time. To prevent this, the error caused by predicting a positive label as negative should be penalized more heavily to enforce the model learning not only the negative samples but also the positive samples as well as possible. The mathematical way of introducing such weighting can be seen below:

$$h(y_j, \hat{y}_j, c_j) = -[p_{cj} y_j log\hat{y}_j + (1 - y_j) \, log \, (1 - \hat{y}_j)] \tag{3.6}$$

$$H(y, \hat{y}) = \frac{1}{N} \sum_{j=1}^{N} h(y_j, \hat{y}_j, c_j), \tag{3.7}$$

where y_j is the true label, $\hat{y}$ is the prediction result (i.e., output of the Sigmoid), c_j is the class, p_{cj} is the weight for class c_j to enforce the learning of the positive samples and N is the number of samples. The class weights are calculated by the negative sample count over the positive sample count for each class, so the prediction errors of the positive samples are as important as the prediction errors of the negative samples.

3.2.3.2. Focal Loss. Focal Loss is first introduced by [48] as a concept to solve the problem of class imbalance, where one or more classes are over-represented, and the other classes are shadowed during the training process, so the model focuses mainly on the most abundant classes. Focal Loss introduces a hyperparameter called the relaxation parameter, γ, which scales the errors of the severely misclassified samples to force the model to focus on the more difficult samples rather than the easier ones.

The vanilla version of focal loss is:

$$H(y, \hat{y}) = -\frac{1}{N} \sum_{j=1}^{N} [y_j log\hat{y}_j \, (1 - \hat{y}_j)^{\gamma} + (1 - y_j) \, log \, (1 - \hat{y}_j) \, \hat{y}_j^{\gamma}], \tag{3.8}$$

where y is the true labels, $\hat{y}$ is the prediction results (i.e., the output of the Sigmoid), N is the number of samples, and γ is the relaxation parameter. This is yet another loss function applied in this thesis.

In our case there is also an imbalance in terms of the label polarity in addition to the class imbalance. To deal with both, like in the binary cross entropy loss, an additional scale factor is introduced to penalize the misclassification of positive samples more heavily:

$$h(y_j, \hat{y}_j, c_j) = -[p_{c_j} y_j log\hat{y}_j \, (1-\hat{y}_j)^{\gamma} + (1-y_j) \, log \, (1-\hat{y}_j) \, \hat{y}_j^{\gamma}] \tag{3.9}$$

$$H(y, \hat{y}) = \frac{1}{N} \sum_{j=1}^{N} h(y_j, \hat{y}_j, c_j), \tag{3.10}$$

where y_j is the true label, $\hat{y}$ is the prediction result (i.e., output of the Sigmoid), c_j is the class, p_{c_j} is the weight for class c_j to enforce the learning of the positive samples, N is the number of samples and γ is the relaxation parameter. Similar to the case of Binary Cross Entropy Loss, class weights are calculated by the negative sample count over the positive sample count for each class.

3.3. Evaluation

3.3.1. Benchmarking metrics

The quality assessments and performance measurements of the models are done by using the following standard criteria.

- **Receiver Operating Characteristic (ROC) Curve:** A graph showing the performance of a classifier. ROC curve compares the True Positive Rate (TPR) to the False Positive Rate (FPR) at varying thresholds. Lowering the threshold causes more items to be classified as positive, which increases both False Positive and True Positive rates. The performance of a classifier over the whole test set or a single sample can be assessed via ROC curve by comparing the graph to a randomly labeling classifier which is $x = y$ line. If the ROC curve of a classifier approaches this baseline, it can be said that the classifier simply has not learned anything useful and assigns labels almost randomly. Classifiers get better as their

ROC curve approaches the upper left corner, since the best case where TPR is 1 and FPR is 0.

- **Area Under the ROC Curve (AUROC):** A concept directly related to ROC curves. It is described as the area lying under the curves' boundaries and simplifies the comparison of different curves. This is useful when multiple intersecting curves are compared.
- **Accuracy:** A fundamental measurement used to calculate the correctly classified sample rate. It is the number of correctly classified samples (i.e., the sum of True Positives and True Negatives) over the size of the sample set.
- **Precision:** A measure calculated by dividing the number of correctly classified positive samples (i.e., True Positives) by the number of all positively classified samples (i.e., the sum of True Positives and False Positives). This measure indicates how good the model is at correctly predicting the positive samples while sacrificing some negative samples by classifying them as positive. If a model does not classify a negative sample as positive (i.e., no False Positives), the precision score of the model is 1, which is the best value a model can have.
- **Recall:** A measure calculated by dividing the number of correctly classified positive samples (i.e., True Positives) by the number of all positive samples (i.e., the sum of True Positives and False Negatives). This measure indicates how good the model is at correctly predicting all positive samples. If a model does not classify a positive sample as negative (i.e., no False Negative), the recall score of the model is 1, which is the best value a model can have.
- **F1 Score:** A more robust measure calculated by taking the harmonic mean of the precision and recall scores to simultaneously penalize both negative and positive misclassification of samples.
- **$\mathbf{F_{max}}$ Score:** Maximum of all the F1 scores of individual labels. This is added to provide a fair comparison as the state-of-the-art model, DeepFRI, reports the results with this metric.
- **Area Under the Precision-Recall Curve (AUPR):** A similar concept to AUROC, but this is calculated using the Precision-Recall curve instead of the ROC curve. This metric is also used by DeepFRI.

4. EXPERIMENTS AND RESULTS

4.1. Experimental Setup

The specifications of the computer used to train, validate, test and evaluate the models can be found in Table 4.1.

Table 4.1. Specifications of the computer.

Hardware/Software	**Details**
CPU	Intel® Core™ i7-9700F 8 Core CPU @ 3.00GHz
RAM	4 x Kingston® 32GB DDR4 2667MHz
GPU	Nvidia® GeForce® RTX 2080 Ti
SSD	Samsung® 980 1TB M.2 NVMe
OS	Ubuntu 18.04.4 (Bionic Beaver) LTS

Information about the programming language, and the most important libraries and frameworks used can be found in Table 4.2.

Table 4.2. Programming language, its libraries/frameworks and their versions.

Programming Language / Library / Framework	**Version**
Python	v3.6.9
Transformers	v4.15.0
Pytorch	v1.10.1
Pytorch-Lightning	v1.5.8
Tensorflow	v2.7.0
Pandas	v1.3.5
Numpy	v1.22.0
Scikit-learn	v1.0.2

4.2. Model Hyperparameters

For *Prot-BERT* fine-tuning approach, there are some hyperparameters (i.e., parameters of the model itself) that must be set before the fine-tuning is initialized. To achieve the best performance, these hyperparameters were tuned using a grid search over the reasonable ranges/options. The most important hyperparameters with their grid search options can be found in Table 4.3.

Table 4.3. Hyperparameters and their options for Prot-BERT.

Hyperparameter	**Options**	**Notes**
Loss Function	Binary Cross Entropy	*BCEWithLogitsLoss* in Pytorch
	Weighted Binary Cross Entropy	Positive sample weight enforcement. *BCEWithLogitsLoss* with *pos_weight* parameter in Pytorch
	Focal Loss	Not natively implemented in Pytorch
	Weighted Focal Loss	Positive sample weight enforcement. Not natively implemented in Pytorch
Epoch	[1, 3, 5, 10]	Early Stopping with 2 epoch patience
Optimizer Learning Rate	[$1e^{-6}$, $1e^{-7}$, $1e^{-8}$]	AdamW (Weighted-decaying Adam Optimizer) is the optimizer
Maximum Sequence Length	[256, 512, 1024]	1024 did not work due to incapable hardware
Classification Head Layer Count	[1, 2, 3, 4]	-
Classification Head Activation Function	ReLU	-
Classification Head Hidden Node Count	[100, 200, 500, 1000]	-
Classification Head Dropout Rate	[0, 0.1, 0.2, 0.5]	-

Similar to *Prot-BERT* model, *Text-GCN* model has some hyperparameters to be tuned to get the best out of it. The main ones with their options can be seen in Table 4.4.

Table 4.4. Hyperparameters and their options of Text-GCN.

Hyperparameter	**Options**	**Notes**
Loss Function	Binary Cross Entropy	*sigmoid_cross_entropy_with_logits* in Tensorflow
	Weighted Binary Cross Entropy	Positive sample weight enforcement. *weighted_cross_entropy_with_logits* with *pos_weight* parameter in Tensorflow
	Focal Loss	Not natively implemented in Tensorflow
	Weighted Focal Loss	Positive sample weight enforcement. Not natively implemented in Tensorflow
Epoch	[100, 200, 300]	Early Stopping with 10 epoch patience
Optimizer Learning Rate	[0.001, 0.002, 0.005, 0.01, 0.02, 0.05]	AdamW (Weighted-decaying Adam Optimizer) is the optimizer
Window Size for PMI	[3, 5, 20, 50, 100]	-
Layer Count	[1, 2, 3, 4]	-
Activation Function	ReLU	-
Hidden Node Count	[100, 200, 300, 500, 1000]	-
Dropout Rate	[0, 0.1, 0.2, 0.5]	-

4.3. Experiments and Results

4.3.1. Hyperparameter Analysis

Accuracy as a metric provides better insight as classes become more balanced. In a dataset with a highly imbalanced class distribution, as in our case, the results could be very good in terms of accuracy, while the model is simply predicting the easiest choice, suggesting that no useful learning has taken place. F1 score is considered as a better measurement tool in such cases. Thus, both measurements for both models are reported in this thesis.

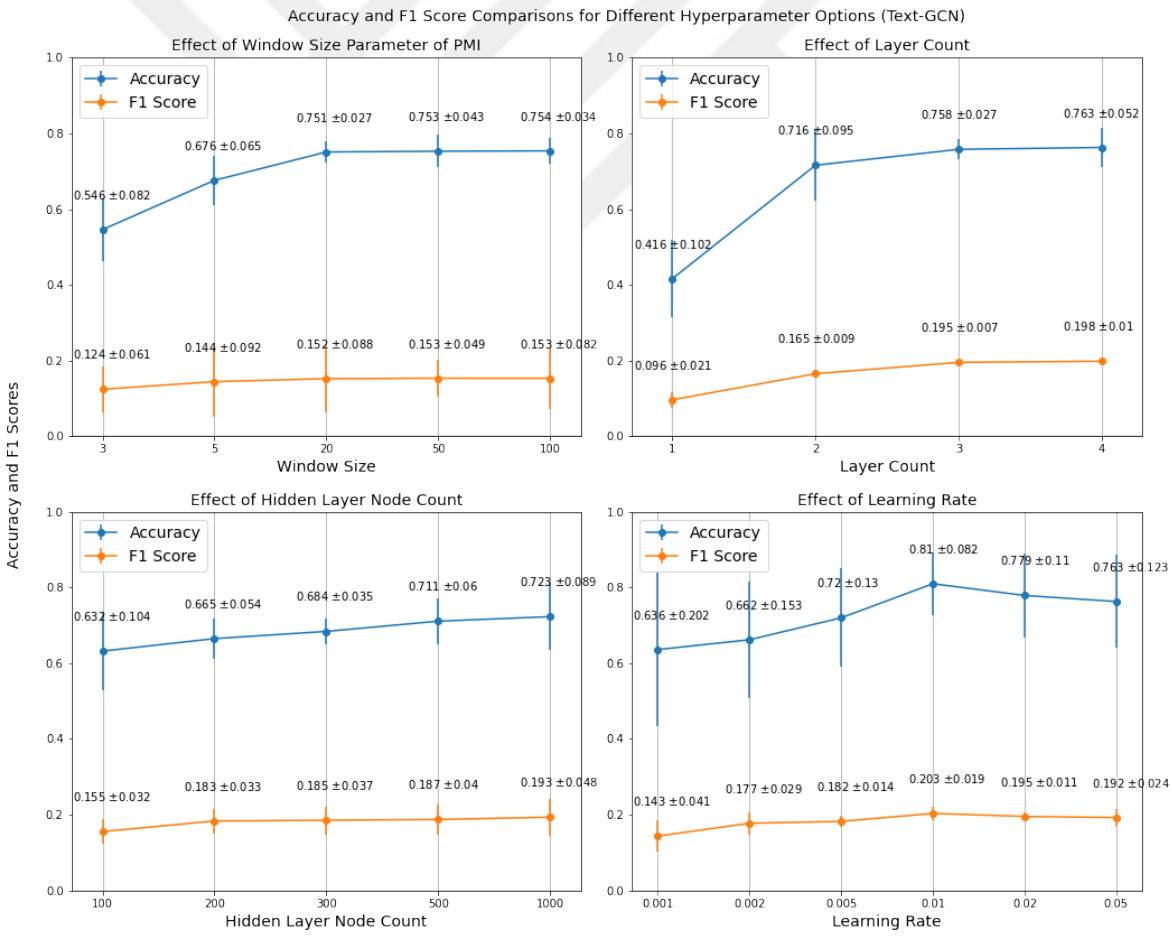

Figure 4.1. The hyperparameters affecting performance the most and their success for different options for Text-GCN.

The effects of various hyperparameters on predictive success were analyzed to find the best ones. The most influential hyperparameters and their effects on accuracy and F1 score are shown in Figures 4.1 and 4.2. Some of the hyperparameters are not included to simplify the plots as their impacts are minimal. One of the most effective common hyperparameter was the choice of loss function, and its effect is shown in Figure B.1 in *Appendix*. The error bars in each figure correspond to the standard deviation of the mean and were calculated over 100 bootstraps of the test set in all experiments.

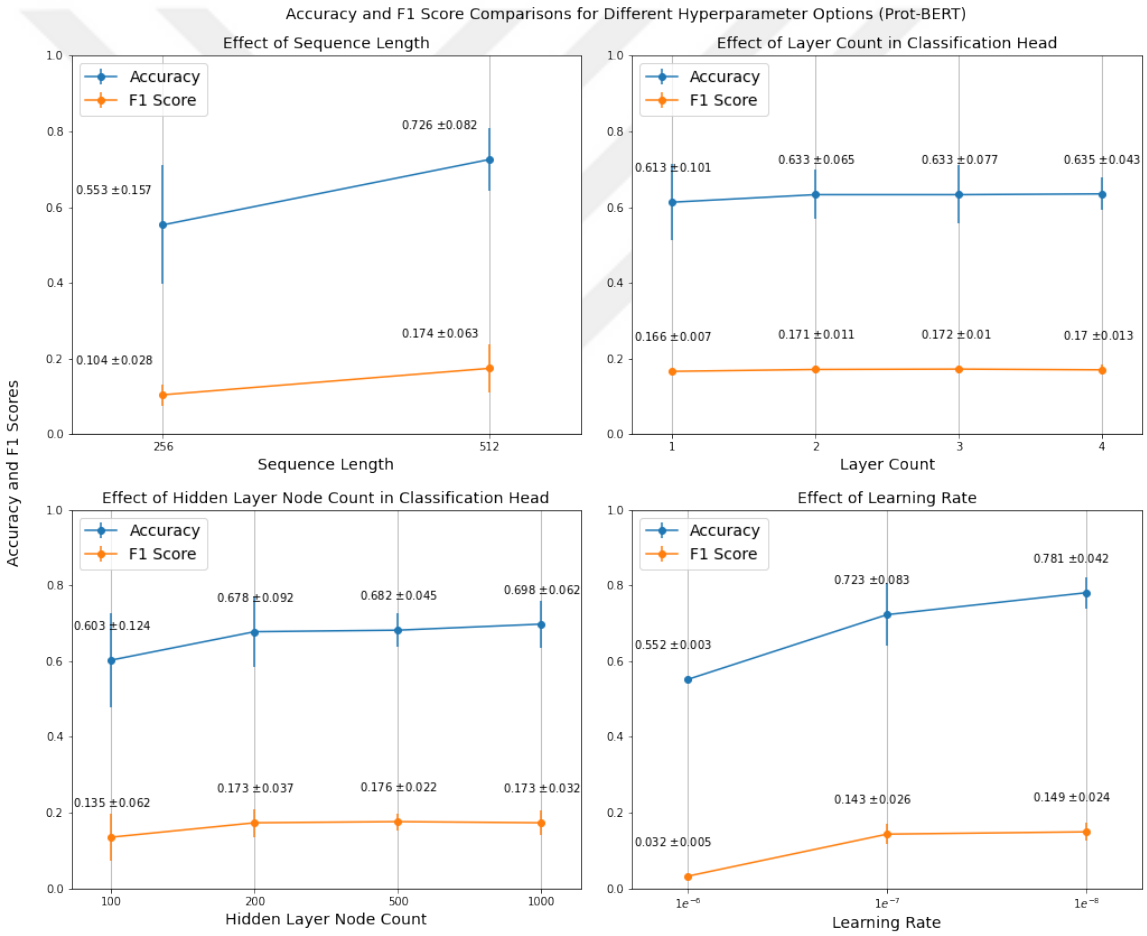

Figure 4.2. The hyperparameters affecting performance the most and their success for different options for Prot-BERT.

Our analysis shows that some hyperparameter options are better than others because they improve the performance significantly. For instance, the *window size* used in PMI calculations for Text-GCN makes the model more effective when increased to a certain point (i.e., 20). If this size is decreased, it is expected that the pair-wise

relationships cannot be adequately covered resulting in poorer performance. Another important hyperparameter that affects success is *layer count*. A three-layer GCN outperforms the one- and two-layer GCNs, and it is found that adding more layers does not significantly improve performance, which is also mentioned in related papers [24, 46]. Moreover, each additional layer doubles the training time. Similarly, the performance is strongly affected by *learning rate* and the best option turned out to be 0.01 for Text-GCN and $1e^{-8}$ for Prot-BERT.

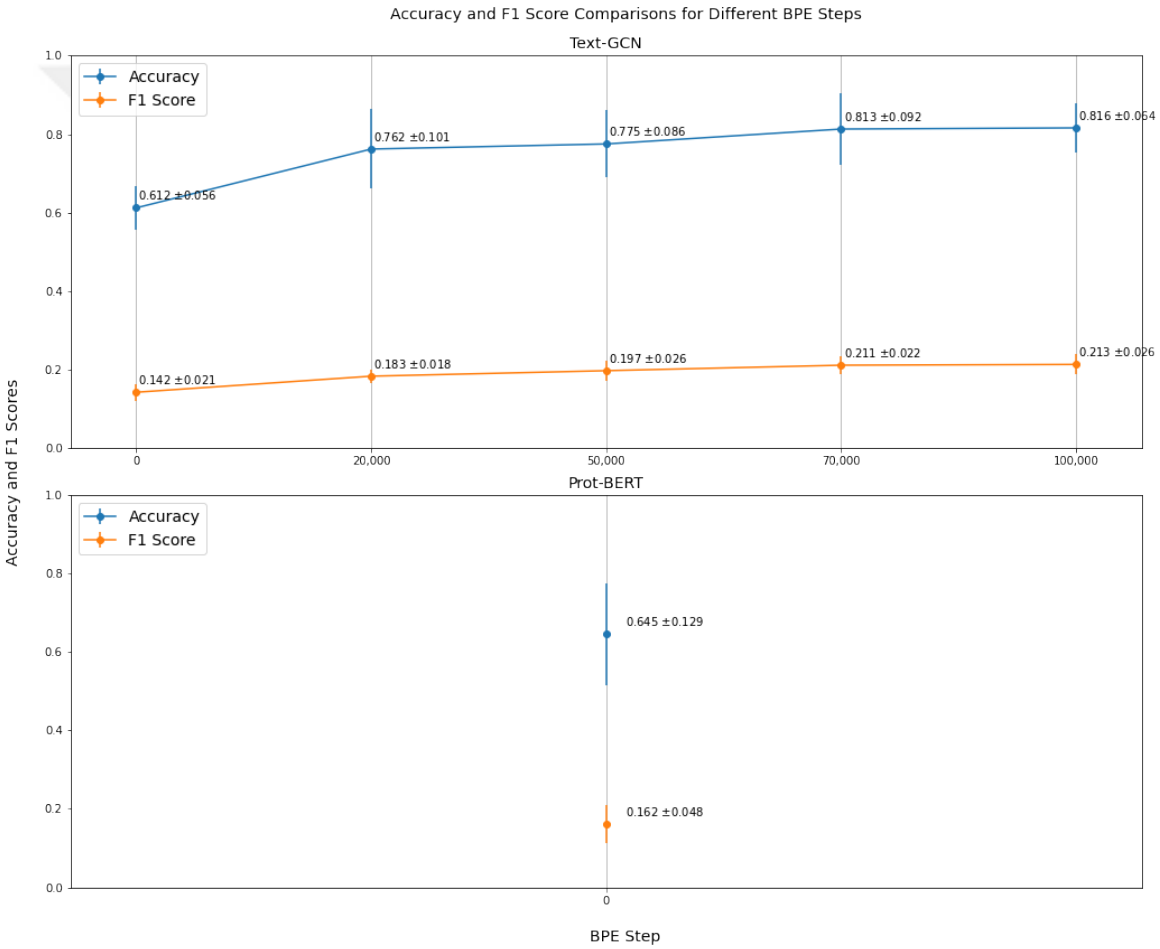

Figure 4.3. The effect of BPE tokenization in terms of accuracy and F1 scores.

Once the hyperparameters are fixed for both models, the *Step* parameter of BPE tokenization was also evaluated. Here, one step corresponds to a new group of amino acids. For example, a BPE with *20,000* steps means that this grouping has been done *20,000* times, so there are that many unique words in the corpus. Since the tokenized protein representations cannot be used with Prot-BERT as mentioned earlier, proteins

are represented with the individual amino acids instead of the the grouped ones for this model. The results in Figure 4.3 show that tokenization of the proteins effectively increases performance up to a certain sweet spot which is between *70,000* and *100,000* steps. Further tokenization would increase the performance, but its rate becomes progressively lower after this point. After all these hyperparameter analyses, they are fixed as in Table 4.5.

Table 4.5. Fixed hyperparameters for the models.

Model	**Hyperparameter**	**Selected Option**
Text-GCN	Loss Function	Weighted Focal Loss
Text-GCN	Epoch	200
Text-GCN	Learning Rate	0.01
Text-GCN	Window Size	20
Text-GCN	Layer Count	3
Text-GCN	Activation Function	ReLU
Text-GCN	Hidden Node Count	500
Text-GCN	Dropout Rate	0.2
Prot-BERT	Loss Function	Weighted Focal Loss
Prot-BERT	Epoch	5
Prot-BERT	Learning Rate	$1e^{-8}$
Prot-BERT	Maximum Sequence Length	512
Prot-BERT	Layer Count	2
Prot-BERT	Activation Function	ReLU
Prot-BERT	Hidden Node Count	1000
Prot-BERT	Dropout Rate	0.1
BPE	Step	70,000

4.3.2. Comparison Results

The models are compared to each other in various aspects, and to the current state-of-the model and its competitors to evaluate the overall performance. The performances of these models are taken from the DeepFRI's article [25].

Table 4.6. Performance of both models and its comparison to the state-of-the-art DeepFRI model along with other competitive models.

Class Type	**Model**	**F_{max} Score ± Std**	**AUPR ± Std**
MF	Text-GCN	0.48 ± 0.01	0.37 ± 0.01
	Prot-BERT	0.42 ± 0.01	0.33 ± 0.02
	DeepFRI	**0.54 ± 0.01**	**0.43 ± 0.01**
	BLAST	0.32 ± 0.02	0.18 ± 0.02
	FFPred	0.23 ± 0.01	0.03 ± 0.01
	DeepGO	0.48 ± 0.01	0.31 ± 0.02
	FunFams	0.51 ± 0.02	0.33 ± 0.03
CC	Text-GCN	0.58 ± 0.01	0.24 ± 0.01
	Prot-BERT	0.51 ± 0.01	0.23 ± 0.01
	DeepFRI	**0.59 ± 0.01**	**0.25 ± 0.02**
	BLAST	0.46 ± 0.01	0.11 ± 0.01
	FFPred	0.49 ± 0.01	0.05 ± 0.01
	DeepGO	0.58 ± 0.01	0.22 ± 0.02
	FunFams	0.58 ± 0.01	0.24 ± 0.02
BP	Text-GCN	0.48 ± 0.01	0.21 ± 0.01
	Prot-BERT	0.46 ± 0.01	0.18 ± 0.01
	DeepFRI	**0.50 ± 0.01**	**0.22 ± 0.01**
	BLAST	0.34 ± 0.01	0.08 ± 0.01
	FFPred	0.34 ± 0.01	0.03 ± 0.01
	DeepGO	0.47 ± 0.01	0.15 ± 0.01
	FunFams	0.47 ± 0.01	**0.22 ± 0.02**

Overall, the models trained on the combined datasets *SWISS+PDB* performed the best, which was expected since this dataset is the largest in size. This is also the dataset selected by DeepFRI, so only this dataset was used for model comparison.

In terms of F_{max} Score, the best model is DeepFRI for all MF, CC and BP class types, 0.54, 0.59 and 0.50 on average respectively, which is expected since it uses additional three-dimensional spatial information directly related to protein functionalities. Text-GCN results in competitive F_{max} Scores, 0.48, 0.58 and 0.48 respectively, for the same class types without use of spatial information. Since F_{max} Score corresponds to the maximum of all F1 scores of individual labels, these results indicate that having spatial information of the proteins increases the predictive performance at least for the labels predicted with highest success. On the other hand, the Prot-BERT model does not achieve competitive performance. Besides the DeepFRI and Text-GCN models, the FunFams and DeepGO models also outperform the Prot-BERT model in all cases in terms of F_{max} Score.

In order to compare the general performance on all functions in a class type, AUPR scores should be inspected instead of F_{max} Score. AUPR is a relatively more important scoring scheme compared to the F_{max} Score because the datasets are extremely sparse and the overall success of a model in such cases can be better measured with AUPR scoring. Therefore, these two scoring techniques should be considered together. DeepFRI is still the best among all models when both scoring techniques are considered together. The only exception is the BP class type where FunFams results in equally successful predictions which is not surprising as the number of classes in the BP case, 1943, is too high for a Deep Learning algorithm. FunFams compares a sequence to the known ones statistically and transfers their functions if they are similar enough. The negative factors like the problem type (i.e., multi-label classification), the number of classes (i.e., 1943) and the sparsity level (i.e., 99.98%) make this statistical approach even more powerful against the Deep Learning approaches. This is evaluated as a not surprising outcome in the DeepFRI article [25].

Table 4.7 compares the performance of the models proposed for protein function prediction in this thesis. Text-GCN outperforms Prot-BERT in each case. One of the reasons for the low performance of Prot-BERT is that it uses the protein sequences by splitting each amino acid and treating them as separate words. However, the nature of proteins is more complex than this simple splitting strategy, which was demonstrated by BPE analysis. Another reason is the limited hardware we had that forced us to cut the sequences. This probably reduced the performance since the only source of information is the sequences and part of them were not used.

Table 4.7. Performance of the two proposed models compared to each other for each of the functional class types.

Class Type	**Model**	**Accuracy ± Std**	**F1 Score ± Std**
MF	Text-GCN	**0.89 ± 0.06**	**0.28 ± 0.03**
	Prot-BERT	0.82 ± 0.04	0.23 ± 0.06
CC	Text-GCN	**0.95 ± 0.02**	**0.21 ± 0.02**
	Prot-BERT	0.86 ± 0.06	0.19 ± 0.03
BP	Text-GCN	**0.92 ± 0.02**	**0.29 ± 0.04**
	Prot-BERT	0.87 ± 0.05	0.24 ± 0.06

To understand the individual performances of the models on different proteins, some samples were selected and their ROC curves were plotted. Specifically, some of the PDB sequences in the test set whose MF-related functionalities are predicted better or worse than the average were selected. Their ROC curves and corresponding AUROCs generated from Text-GCN results can be seen in Figure 4.4. The curves show that the predicted functionalities of the proteins with PDB IDs *1UWY-A* and *1ZI7-A* do not overlap with the actual functionalities, indicating that the model does not work well at least for the functionalities associated with these proteins. The AUROC results suggest that a model that randomly assigns functionalities would perform equally or even better than the Text-GCN model for the functionalities these specific proteins have.

On the other hand, the predictions for the proteins with PDB-IDs *1CNT-1*, *1DI7-A*, *1E0T-A* and *1QMT-A* are substantially correct because the curves are close to the upper left corner. The AUROC scores of these predictions are at least 0.8. This can be interpreted as Text-GCN predicts the functionalities associated with these proteins much better than a random classifier.

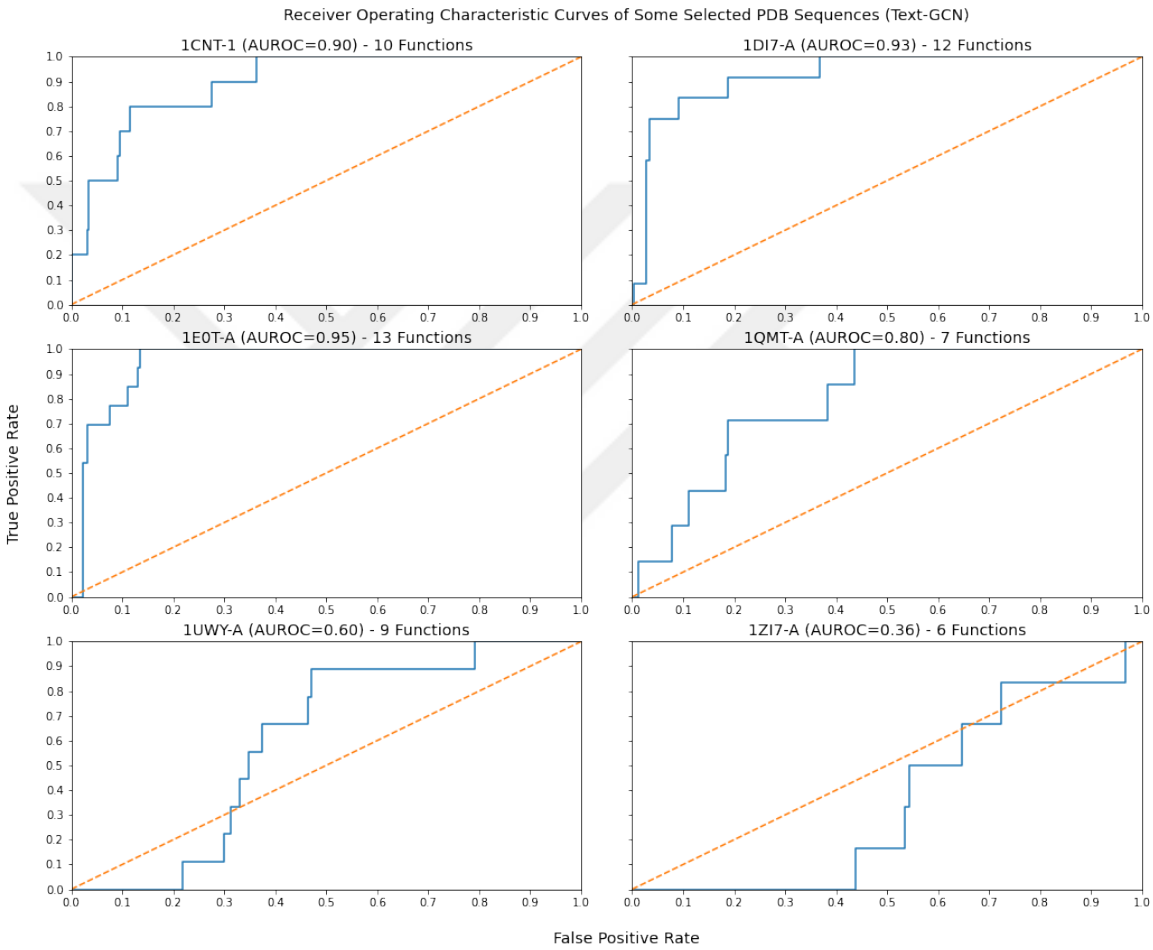

Figure 4.4. Text-GCN ROC curves and areas under them for the proteins with PDB-IDs *1CNT-1*, *1DI7-A*, *1E0T-A*, *1QMT-A*, *1UWY-A* and *1ZI7-A*.

These proteins were selected intentionally to show that the Text-GCN model is strong enough to predict some of the functionalities, while it also has some weaknesses. The ROC curves of some randomly selected proteins from the test set can be seen in Figure 4.5.

Additional ROC curves comparing sample results of different cases where two models compete can be seen in Figures B.2, B.3, B.4 and B.5 in *Appendix*. This indicates that even though the Text-GCN model outperforms the Prot-BERT model on average, some of the labels are predicted better by the Prot-BERT model.

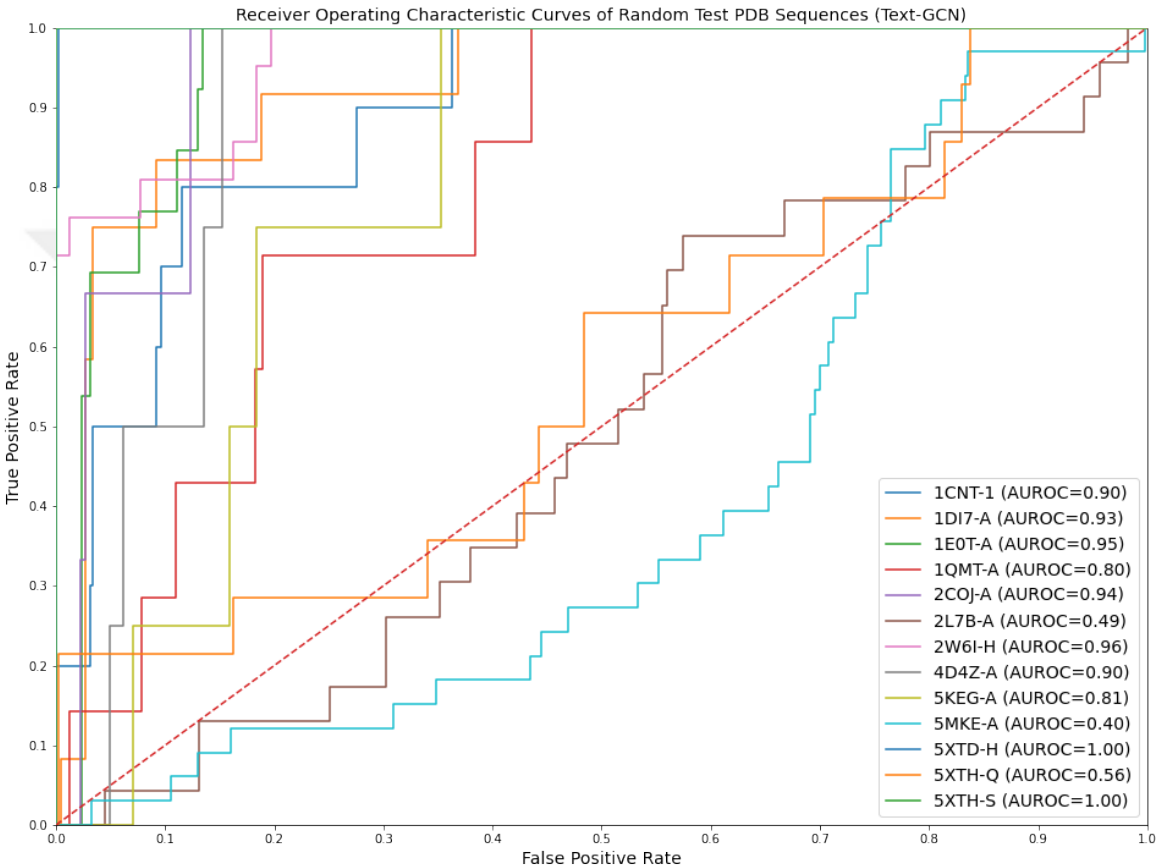

Figure 4.5. Text-GCN ROC curves and areas under them for the randomly sampled proteins.

One of the immediate conclusions would be the effect of sparsity of the labels, as this is probably the reason for the weak performance for certain functions. To test this assumption, F1 scores of each of the labels were calculated for each label and then sorted based on the number of samples associated with these functions in ascending order.

Figure 4.6 shows the relationship between the degrees of sparsity and the F1 scores, indicating that the assumption of sparsity-success inverse relationship holds for the majority of the classes being predicted incorrectly. It is clear that the classes

with the weakest performance are generally also those with the greatest imbalance in polarity. The left-most label in this plot, (GO:001684), is associated with only 1 protein out of 3415 proteins in the test set and 46 proteins out of 24961 proteins for PDB dataset. This is also true for the SWISS dataset where 472 proteins out of 202807 proteins are associated with this function. This figure shows the analysis of MF class type, and the same analysis was performed with the results of other class types for both models. It was found that a similar trend holds for those cases as well, as shown in Figures C.1, C.2, C.3, D.1 and D.2 in *Appendix*.

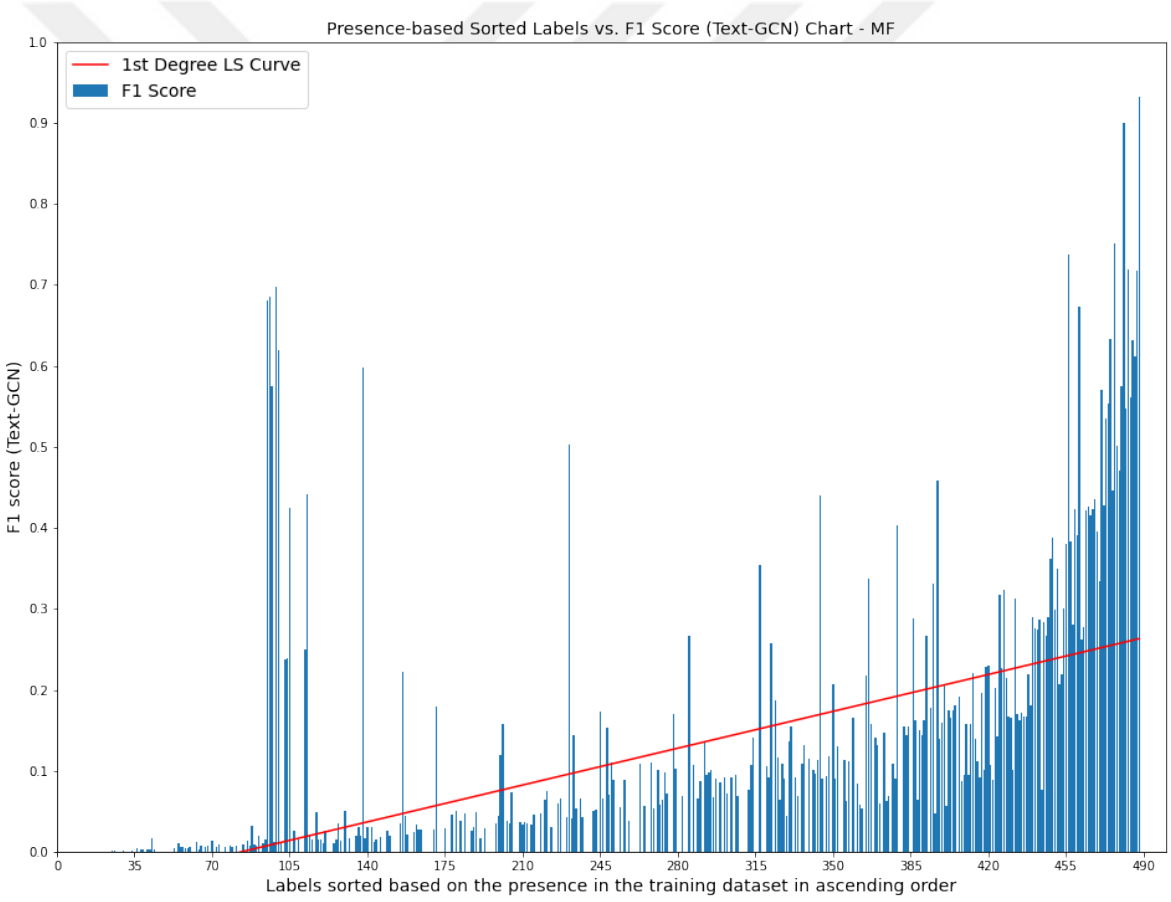

Figure 4.6. Text-GCN F1 scores of each label in MF ordered by their presence in ascending order.

To test the effect of sparsity, another experiment was done as shown in Table 4.8. Here, the dataset is manipulated to take (i) the subset of the data that refers to the 100 most frequent labels, and (ii) the subset of the data that refers to the 100 labels with average frequency. Then these two subsets were used to train and test both models.

The results in the table clearly show that using the most frequent labels increases the performances in all cases, indicating that the sparsity is one of the main problems here. Even for the subset (i), the sparsity level is 98.52% on average, which is still extremely high compared to regular machine learning use cases. The F1 scores show that even the subset (i) is still so sparse that the models barely perform better.

Table 4.8. The results of sparsity experiment with two different subsets.

Class Type	**Model**	**Subset**	**Accuracy ± Std**	**F1 Score ± Std**
MF	Text-GCN	Most Abundant 100 Labels	**0.852 ± 0.043**	**0.247 ± 0.014**
		100 Labels with Average Abundance	0.832 ± 0.083	0.225 ± 0.024
	Prot-BERT	Most Abundant 100 Labels	0.817 ± 0.036	0.228 ± 0.011
		100 Labels with Average Abundance	0.786 ± 0.043	0.205 ± 0.023
CC	Text-GCN	Most Abundant 100 Labels	**0.872 ± 0.033**	**0.284 ± 0.045**
		100 Labels with Average Abundance	0.844 ± 0.052	0.253 ± 0.057
	Prot-BERT	Most Abundant 100 Labels	0.826 ± 0.035	0.239 ± 0.032
		100 Labels with Average Abundance	0.795 ± 0.042	0.214 ± 0.035
BP	Text-GCN	Most Abundant 100 Labels	**0.862 ± 0.041**	**0.326 ± 0.073**
		100 Labels with Average Abundance	0.832 ± 0.055	0.288 ± 0.063
	Prot-BERT	Most Abundant 100 Labels	0.840 ± 0.033	0.262 ± 0.069
		100 Labels with Average Abundance	0.816 ± 0.058	0.230 ± 0.073

Another important analysis is done to understand the effect of sequence clipping. In the case of Prot-BERT, the sequences had to be clipped due to the incapable hardware. The sequences are cut if they exceed 512 amino acids, and only the first part containing 512 amino acids is used. This is controlled by the hyperparameter named *Maximum Sequence Length*. Thus, some of the information is lost for some of the sequences. This is, as mentioned, one of the issues with the Prot-BERT model. To point the information lost, Figure 4.7 can be inspected. Here, a cumulative histogram of the sequence lengths is shown for each class type separately. The point corresponding to 512 is annotated in each plot.

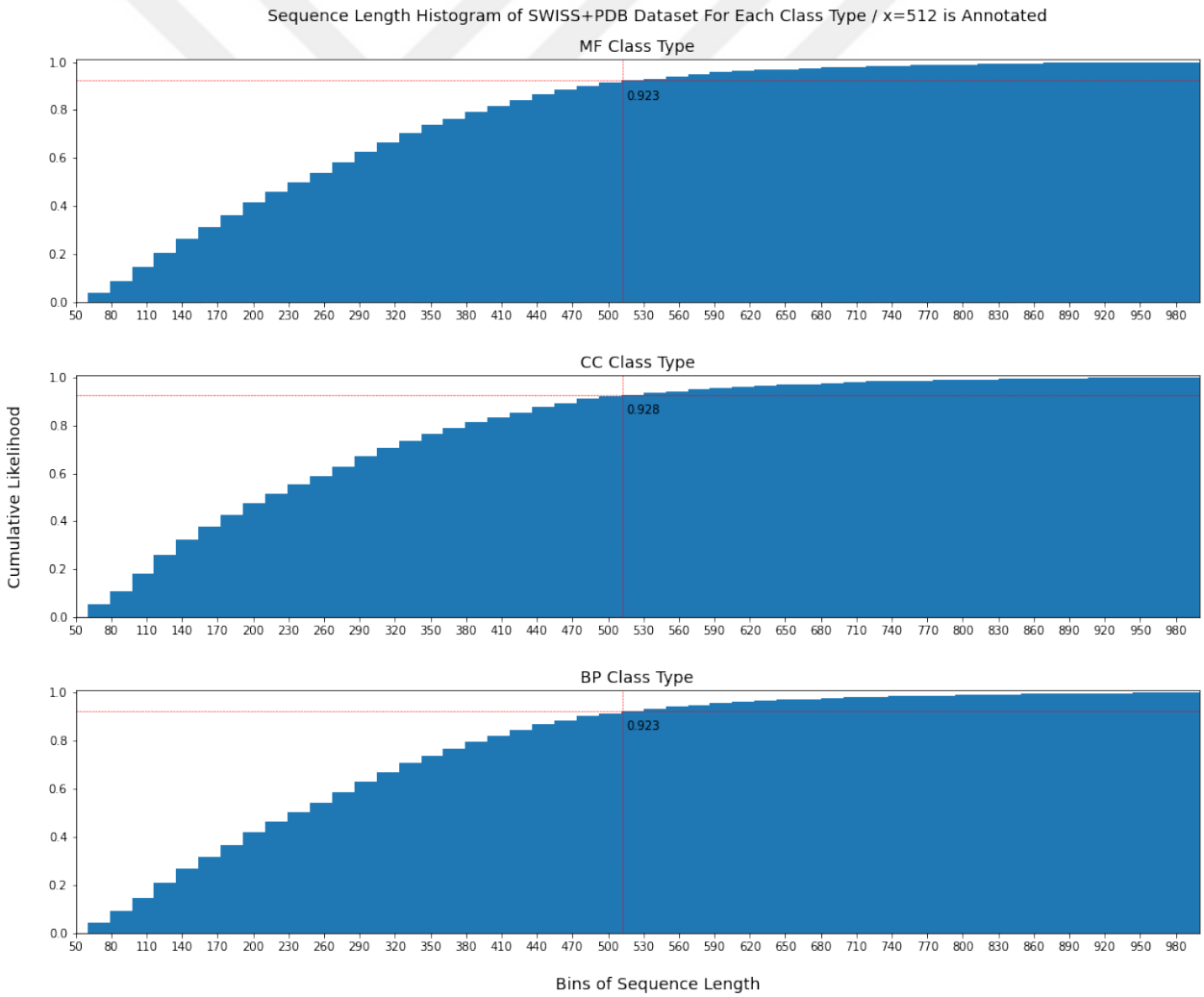

Figure 4.7. Histogram showing the cumulative sequence length distribution.

These plots point out that around 93% of the sequences are not clipped at all. The remaining 7% of the sequences are clipped and at least one amino acid of these sequences is not used. To analyze how much information is lost per sequence, Figure 4.8 can be inspected, where the histogram of the sequences longer than 512 amino acids is shown for each case separately.

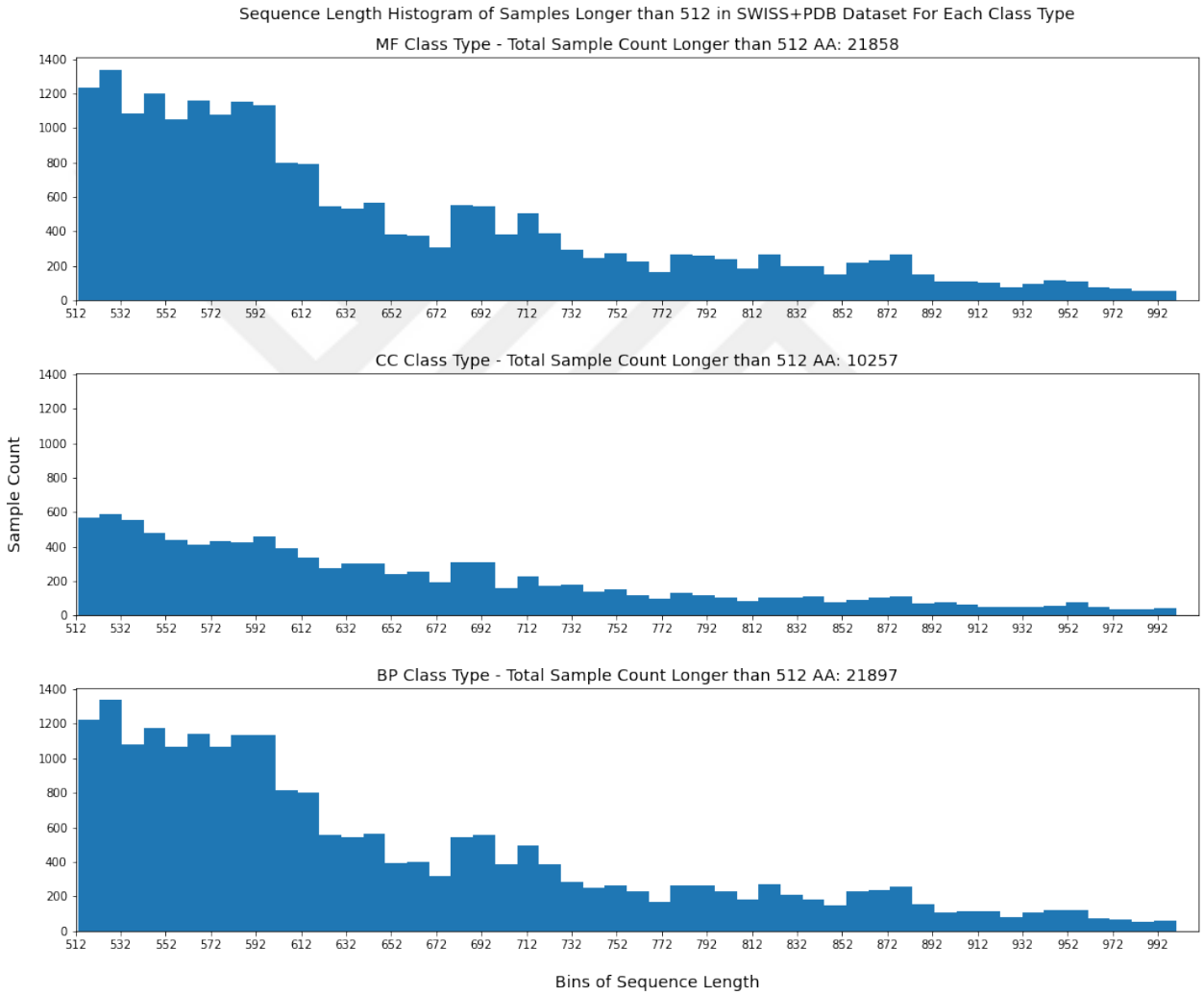

Figure 4.8. Histogram showing the sequence length distribution that are longer than 512 amino acids (AA).

Each of these sequences is clipped to end up with a sequence with 512 amino acids. As the x-axis increases, the lengths of the sequences and the information lost due to the clipping process also increase. To analyze how much information is lost in total, Figure 4.9 can be inspected, where the x-axis corresponds to the sequence length and the y-axis corresponds to the total number of amino acids in the sequences

of these sequence lengths. Here, the blue bar chart represents the amino acids used while the orange one represents the amino acids removed. It can be clearly seen that the information lost due to the clipping process is a small part of the total information available. However, as the sequence length increases, the information lost gets closer to the information used, meaning that almost half of the available information is clipped out in the most extreme cases. This might be one of the reasons behind the relatively lower performance of the Prot-BERT model compared to the Text-GCN model.

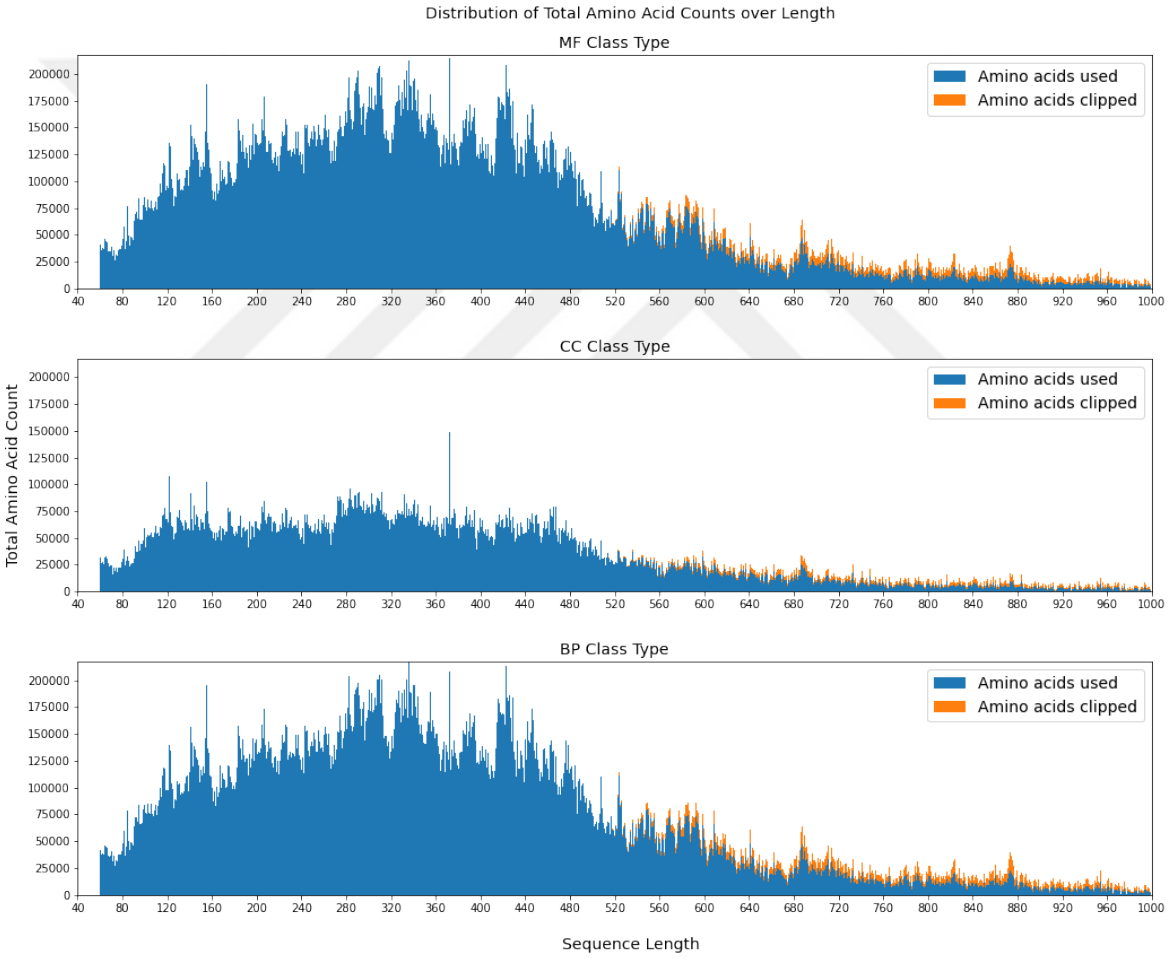

Figure 4.9. Total amino acid count of the samples distributed over their lengths.

To make a fair comparison, Text-GCN is trained and tested one more time with the clipped data. The results are shown in Table 4.9. Compared to the results obtained with the unclipped data, which can be seen in Table 4.7, the performance of Text-GCN decreases because of the loss of information. Even though this is an expected outcome, the Text-GCN model still outperforms the Prot-BERT model. If the Prot-BERT model

could be trained with the grouped amino acids, then the difference might become even smaller.

Table 4.9. Performance of the two proposed models trained and tested with the clipped data for each of the functional class types.

Class Type	**Model**	**Accuracy ± Std**	**F1 Score ± Std**
MF	Text-GCN	**0.85 ± 0.05**	**0.25 ± 0.03**
	Prot-BERT	0.82 ± 0.04	0.23 ± 0.06
CC	Text-GCN	**0.92 ± 0.02**	**0.20 ± 0.03**
	Prot-BERT	0.86 ± 0.06	0.19 ± 0.03
BP	Text-GCN	**0.89 ± 0.03**	**0.26 ± 0.03**
	Prot-BERT	0.87 ± 0.05	0.24 ± 0.06

5. DISCUSSION

Our findings suggest that the functionalities of proteins can be predicted from their amino acid sequences at a certain level, which could be enhanced either by three-dimensional spatial information or by treating the sequences as complex documents with sophisticated relationships between their amino acids. In this thesis, spatial information was absent, so solely the amino acid sequences were used to predict the functionalities.

In [25], it is stated that high-quality sequence alignment is often sufficient to transfer structural information, but the sequence alignments are not sufficient to transfer functional information because different thresholds are required for different functions, partial alignments and protein moonlighting. An evidence of this problem could be the poor performance of BLAST in CAFA [49–51]. This is the main difference between our proposed models that only use protein sequence information and the state-of-the-art DeepFRI model that also uses three-dimensional structural information. Although DeepFRI achieved better performance than our proposed models, the Text-GCN based approach followed it closely.

An important note on hyperparameters is the choice of sequence length in the case of the Prot-BERT model. In this model, the sequences are cut from the beginning with respect to the *Maximum Sequence Length* hyperparameter. It is expected that the performance is affected as the only information source is the amino acid sequence. The main assumption of this thesis is that the functionalities are directly related to the sequence, so using only the first N amino acids would naturally affect the performance. The main reason behind this clipping is the insufficient hardware we had. This is definitely one of the drawbacks of using such a large model.

Another disadvantage of using Prot-BERT was the choice of tokenization by the authors, which cannot be changed later on as this is a pretrained model. The model

did not take advantage of using BPE-tokenized word representations. We expect the use of these word representations to increase the success of the model as in the case of Text-GCN.

It should also be noted that fine-tuning the Prot-BERT model, which is a BERT-large-based model with more than 300M parameters, took at least 10x as much time as training the Text-GCN model from scratch. It also requires more powerful hardware than Text-GCN. For example, experimentation including a higher sequence length could not be done because the VRAM of the GPU available was insufficient. These limitations make the Prot-BERT model even less useful compared to the Text-GCN model. However, with a more powerful computer that has a decent GPU, this model can be trained from scratch, which may result in improved performance. This is one of the future works.

One of the main assumptions in this thesis is that the information hidden in the amino acid sequence is necessary and sufficient to make a functional assessment of proteins even if the predictions are not completely accurate. Even though for some protein function classes sequence information was sufficient for accurate prediction, the results suggested that for majority of the classes three-dimensional information provides additional useful information. Improving the text-based models may lead to better utilisation of protein sequence information, which may result in more accurate protein function prediction.

Changing the way of treating the amino acid sequences would improve predictions because the sequence takes on a complicated shape when folded into its three-dimensional form, so that the amino acids that are separated in the sequence form could be spatially close. Although it was a better choice to use amino acid groups as the building blocks instead of the individual amino acids, a better approach would be to tokenize the sequences non-sequentially by considering the amino acids that are farther apart from each other as a potential group due to the nature of the folding scheme.

However, this approach would increase the overhead of the tokenization process because of the exponentially increasing possible combinations. Handling tokenization in this a way would reduce the need for three-dimensional structures because we would be taking advantage of the true information contained in sequences. This approach can be considered as a future work.

Another problem was the lack of positively labelled data for each of the cases. As mentioned earlier, the classes are extremely sparse (i.e., 99.98% on average). Considering this, it is highly unlikely that a good prediction will be made, as the models tend to overfit to the simplest solution, and almost always predict the negative class. By tuning the hyperparameters and choosing appropriate loss functions, this problem has been reduced, but it is still difficult for the models to learn the abstract information that leads to the positive labels, as these are exceptionally rare compared to a regular classification problem.

The effect of this sparseness can be seen in the experiment conducted with two different subsets of the data, one containing the 100 most frequent labels and the other containing 100 labels with average frequency. There are even some labels having only 30 positive samples in the entire training dataset while they have hundreds of thousands negative samples. It is almost impossible to learn from such a distribution, so that the models have failed on these labels regardless of the hyperparameter choice. Our label-based results cannot be compared to the state-of-the-art model DeepFRI, because the dataset is not discussed from this perspective in the corresponding paper [25] and no results for individual classes were reported.

6. CONCLUSION

In this thesis, we investigated whether using the amino acid sequences alone, without the three-dimensional spatial information, would lead to protein function prediction by using two different first-class language processing methods in combination with a tokenization method. The first language processing method is a GCN adaptation with a heterogeneous representation where the documents (i.e., the proteins) and the words (i.e., the amino acid groups) are the nodes and the relationships between them are the edges. These amino acid groups are obtained by applying the BPE tokenizer to the corpus to find the conserved amino acid subsequences resulting from evolution. The second language processing method is a BERT adaptation pretrained on a very large corpus for the MLM task and modified to perform this downstream task by adding a classification head on top of the BERT structure. We tested these two models on three different functional class types: Molecular Functions, Cellular Components and Biological Processes. We assessed the quality of both the datasets and the models against a number of criteria.

The results show that the GCN-based approach (i.e., Text-GCN) outperforms the BERT-based approach (i.e., Prot-BERT) as Text-GCN yields significantly better scores for each of the three class types. However, the state-of-the-art model in this area achieves better F_{max} and AUPR scores, possibly because it uses the three-dimensional spatial information of the proteins in addition to the sequence itself. The motivation behind this thesis is to achieve similar or equal performance to the state-of-the-art model without this additional source of information, and this is almost the case.

As future work, it is planned to tokenize the sequences non-sequentially to capture more informative subsequences as a protein performs its function in its three-dimensional form, where amino acids separated in one-dimensional form could be close to each other in this form. This may increase performance even though the actual spatial information is still missing. Another future work is to train Prot-BERT from

scratch to use amino acid groups instead of individual amino acids, which could increase the overall performance of the model. Another possible improvement is not to use the models to classify documents, but to embed the words and documents into vectors and use these embeddings with different discriminative models that can better handle the sparse data.

REFERENCES

1. Berg, B., R. Keys, C. Dobson and R. Ellis, "Macromolecular Crowding Perturbs Protein Refolding Kinetics: Implications for Folding Inside The Cell", *The EMBO Journal*, Vol. 19, pp. 3870–5, 09 2000.

2. Ashburner, M., 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. Bairoch, A., "The ENZYME Database in 2000", *Nucleic Acids Research*, Vol. 28, No. 1, pp. 304–305, 2000.

4. Kimura, M. and T. Ohta, "On Some Principles Governing Molecular Evolution", *Proceedings of the National Academy of Sciences*, Vol. 71, No. 7, pp. 2848–2852, 1974.

5. wwPDB consortium, "Protein Data Bank: The Single Global Archive for 3D Macromolecular Structure Data", *Nucleic Acids Research*, Vol. 47, No. D1, pp. D520–D528, 10 2018.

6. Dana, J. M., A. Gutmanas, N. Tyagi, G. Qi, C. O'Donovan, M. Martin and S. Velankar, "SIFTS: Updated Structure Integration with Function, Taxonomy and Sequences Resource Allows 40-fold Increase in Coverage of Structure-based Annotations for Proteins", *Nucleic Acids Research*, Vol. 47, No. D1, pp. D482–D489, 11 2018.

7. Bairoch, A. and R. Apweiler, "The SWISS-PROT Protein Sequence Data Bank and Its New Supplement TREMBL", *Nucleic Acids Research*, Vol. 24, No. 1, pp. 21–25, 01 1996.

8. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser

and I. Polosukhin, "Attention Is All You Need", *Advances in Neural Information Processing Systems*, Vol. 30, 2017.

9. Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *arXiv*, Vol. abs/1810.04805, 2018.

10. Elnaggar, A., M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik and B. Rost, "ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2020.

11. Rogers, A., O. Kovaleva and A. Rumshisky, "A Primer in BERTology: What We Know About How BERT Works", *Transactions of the Association for Computational Linguistics*, Vol. 8, pp. 842–866, 2021.

12. Ardejani, M. S., E. T. Powers and J. W. Kelly, "Using Cooperatively Folded Peptides to Measure Interaction Energies and Conformational Propensities", *Accounts of Chemical Research*, Vol. 50, No. 8, pp. 1875–1882, 2017.

13. Rodwell, V. W. and R. K. Murray, *Harper's Illustrated Biochemistry, 31e*, McGraw-Hill Education, 2018.

14. Gage, P., "A New Algorithm for Data Compression", *The C Users Journal Archive*, Vol. 12, pp. 23–38, 1994.

15. Li, X. and D. Fourches, "SMILES Pair Encoding: A Data-Driven Substructure Tokenization Algorithm for Deep Learning", *Journal of Chemical Information and Modeling*, Vol. 61, No. 4, pp. 1560–1569, 2021.

16. Kawano, K., S. Koide and C. Imamura, "Seq2seq Fingerprint with Byte-Pair Encoding for Predicting Changes in Protein Stability upon Single Point Mutation",

IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 17, No. 5, pp. 1762–1772, 2019.

17. Asgari, E., A. C. McHardy and M. R. Mofrad, "Probabilistic Variable-length Segmentation of Protein Sequences for Discriminative Motif Discovery (DiMotif) and Sequence Embedding (ProtVecX)", *Scientific reports*, Vol. 9, No. 1, pp. 1–16, 2019.

18. Wallis, W. D., *A Beginner's Guide to Graph Theory*, Springer Science & Business Media, 2007.

19. Duvenaud, D., D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik and R. P. Adams, "Convolutional Networks on Graphs for Learning Molecular Fingerprints", *arXiv*, Vol. abs/1509.09292, 2015.

20. Coley, C. W., R. Barzilay, W. H. Green, T. S. Jaakkola and K. F. Jensen, "Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction", *Journal of Chemical Information and Modeling*, Vol. 57, No. 8, pp. 1757–1772, 2017.

21. Somnath, V. R., C. Bunne and A. Krause, "Multi-scale Representation Learning on Proteins", *Advances in Neural Information Processing Systems*, Vol. 34, 2021.

22. Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-Cam: Visual Explanations from Deep Networks via Gradient-based Localization", *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

23. Defferrard, M., X. Bresson and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering", *Advances in Neural Information Processing Systems*, Vol. 29, pp. 3844–3852, 2016.

24. Kipf, T. N. and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", *arXiv*, Vol. abs/1609.02907, 2016.

25. Gligorijević, V., P. D. Renfrew, T. Kosciolek, J. K. Leman, D. Berenberg, T. Vatanen, C. Chandler, B. C. Taylor, I. M. Fisk, H. Vlamakis *et al.*, "Structure-based Protein Function Prediction Using Graph Convolutional Networks", *Nature Communications*, Vol. 12, No. 1, pp. 1–14, 2021.

26. Altschul, S. F., W. Gish, W. Miller, E. W. Myers and D. J. Lipman, "Basic Local Alignment Search Tool", *Journal of Molecular Biology*, Vol. 215, No. 3, pp. 403–410, 1990.

27. Radivojac, P., 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.

28. Eddy, S. R., "A New Generation of Homology Search Tools Based on Probabilistic Inference", *Genome Informatics 2009: Genome Informatics Series*, Vol. 23, pp. 205–211, World Scientific, 2009.

29. Cozzetto, D., F. Minneci, H. Currant and D. T. Jones, "FFPred 3: Feature-based Function Prediction for All Gene Ontology Domains", *Scientific Reports*, Vol. 6, No. 1, pp. 1–11, 2016.

30. Das, S., D. Lee, I. Sillitoe, N. L. Dawson, J. G. Lees and C. A. Orengo, "Functional Classification of CATH Superfamilies: a Domain-based Approach for Protein Function Annotation", *Bioinformatics*, Vol. 31, No. 21, pp. 3460–3467, 2015.

31. Bonetta, R. and G. Valentino, "Machine Learning Techniques for Protein Function Prediction", *Proteins: Structure, Function, and Bioinformatics*, Vol. 88, No. 3, pp. 397–413, 2020.

32. Fa, R., D. Cozzetto, C. Wan and D. T. Jones, "Predicting Human Protein Function with Multi-task Deep Neural Networks", *PloS One*, Vol. 13, No. 6, p. e0198216,

2018.

33. Rifaioglu, A. S., T. Doğan, M. J. Martin, R. Cetin-Atalay and V. Atalay, "DEEPred: Automated Protein Function Prediction with Multi Task Feed Forward Deep Neural Networks", *Scientific Reports*, Vol. 9, No. 1, pp. 1–16, 2019.

34. Bepler, T. and B. Berger, "Learning Protein Sequence Embeddings Using Information from Structure", *arXiv*, Vol. abs/1902.08661, 2019.

35. AlQuraishi, M., "End-to-end Differentiable Learning of Protein Structure", *Cell Systems*, Vol. 8, No. 4, pp. 292–301, 2019.

36. Wang, S., S. Sun, Z. Li, R. Zhang and J. Xu, "Accurate De Novo Prediction of Protein Contact Map by Ultra Deep Learning Model", *PLoS Computational Biology*, Vol. 13, No. 1, p. e1005324, 2017.

37. Kulmanov, M., M. A. Khan and R. Hoehndorf, "DeepGO: Predicting Protein Functions from Sequence and Interactions Using a Deep Ontology-aware Classifier", *Bioinformatics*, Vol. 34, No. 4, pp. 660–668, 2018.

38. Jiménez, J., S. Doerr, G. Martínez-Rosell, A. S. Rose and G. De Fabritiis, "DeepSite: Protein-binding Site Predictor Using 3D-Convolutional Neural Networks", *Bioinformatics*, Vol. 33, No. 19, pp. 3036–3042, 2017.

39. Amidi, A., S. Amidi, D. Vlachakis, V. Megalooikonomou, N. Paragios and E. I. Zacharaki, "EnzyNet: Enzyme Classification Using 3D Convolutional Neural Networks on Spatial Representation", *PeerJ*, Vol. 6, p. e4750, 2018.

40. Henaff, M., J. Bruna and Y. LeCun, "Deep Convolutional Networks on Graph-Structured Data", *arXiv*, Vol. abs/1506.05163, 2015.

41. Fout, A. M., *Protein Interface Prediction Using Graph Convolutional Networks*, Ph.D. Thesis, Colorado State University, 2017.

42. Gelman, S., S. A. Fahlberg, P. Heinzelman, P. A. Romero and A. Gitter, "Neural Networks to Learn Protein Sequence-Function Relationships from Deep Mutational Scanning Data", *BioRxiv*, pp. 2020–10, 2021.

43. Rostlab, *Hugging Face Repository of All Pretrained ProtTrans Models*, 2020, `https://huggingface.co/Rostlab`, accessed in September 2021.

44. Rostlab, *Hugging Face Repository of Pretrained Prot-BERT Model*, 2020, `https://huggingface.co/Rostlab/prot_bert`, accessed in September 2021.

45. Falcon, W. and The PyTorch Lightning team, *PyTorch Lightning*, 2019, `https://github.com/PyTorchLightning/pytorch-lightning`, accessed in September 2021.

46. Yao, L., C. Mao and Y. Luo, "Graph Convolutional Networks for Text Classification", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 7370–7377, 2019.

47. Shannon, C. E., "A Mathematical Theory of Communication", *The Bell System Technical Journal*, Vol. 27, No. 3, pp. 379–423, 1948.

48. Lin, T.-Y., P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection", *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017.

49. Dodge, C., R. Schneider and C. Sander, "The HSSP Database of Protein Structure—Sequence Alignments and Family Profiles", *Nucleic Acids Research*, Vol. 26, No. 1, pp. 313–315, 1998.

50. Radivojac, P., 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.

51. Zhou, N., Y. Jiang, T. R. Bergquist, A. J. Lee, B. Z. Kacsoh, A. W. Crocker, K. A. Lewis, G. Georghiou, H. N. Nguyen, M. N. Hamid *et al.*, "The CAFA Challenge Reports Improved Protein Function Prediction and New Functional Annotations for Hundreds of Genes Through Experimental Screens", *Genome Biology*, Vol. 20, No. 1, pp. 1–23, 2019.

APPENDIX A: ADDITIONAL STATISTICS OF DATASETS

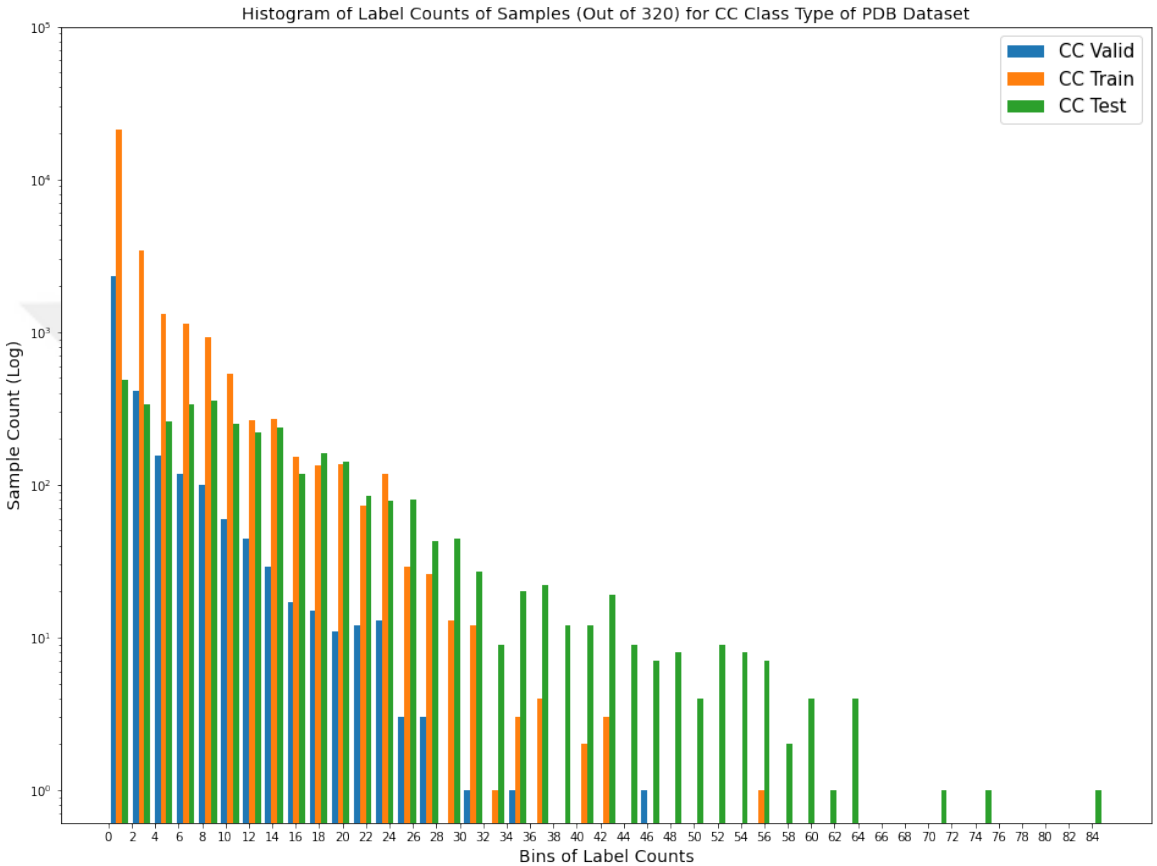

Figure A.1. Histogram of samples in terms of number of functions associated with them for CC class type in PDB dataset.

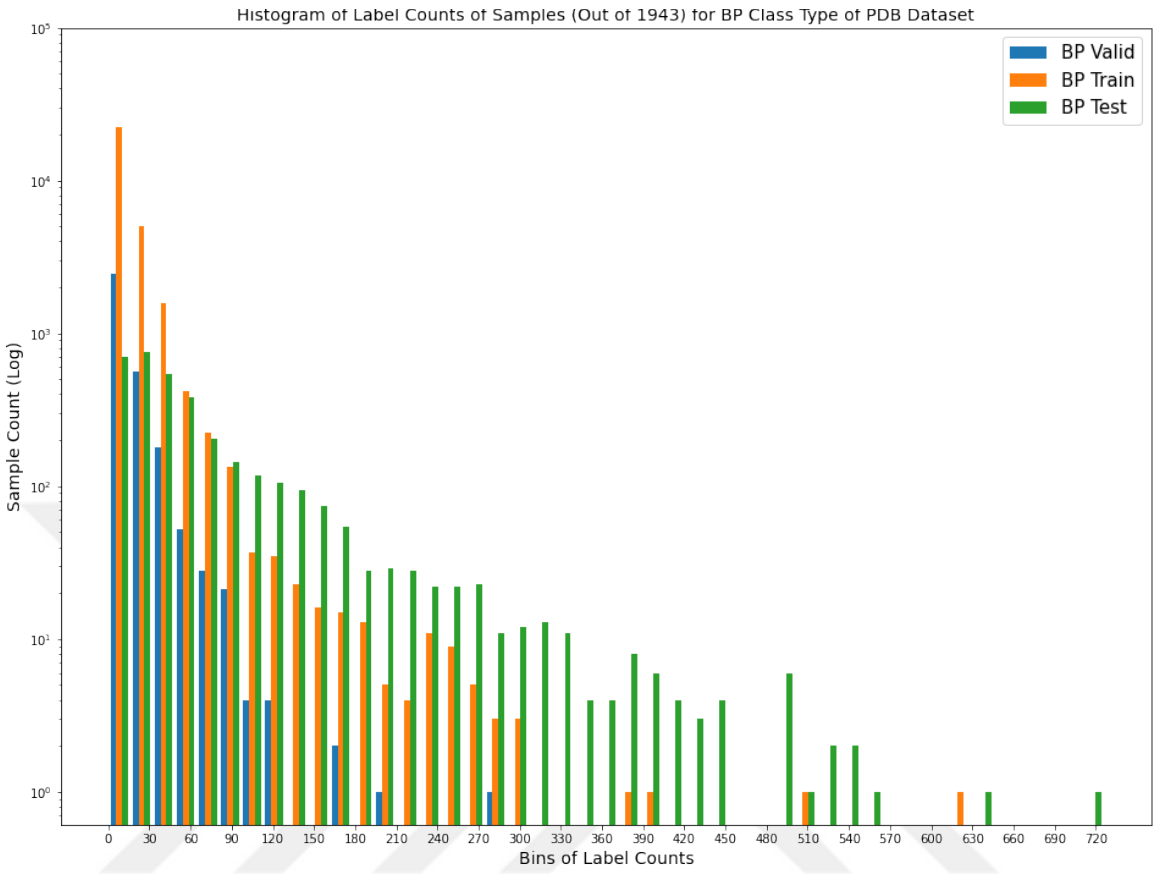

Figure A.2. Histogram of samples in terms of number of functions associated with them for BP class type in PDB dataset.

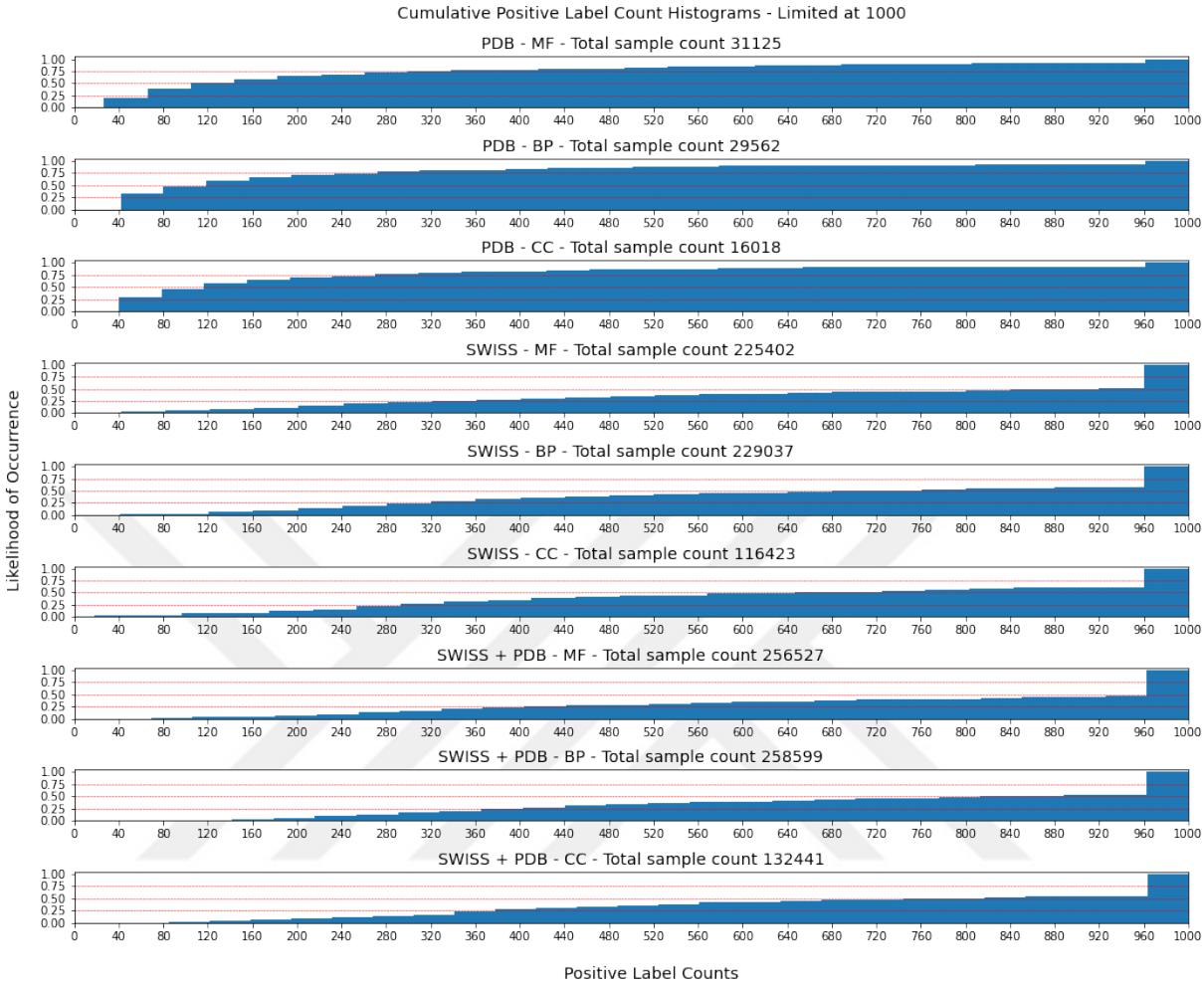

Figure A.3. The same information with Figure 3.2 but limited at 1000 to focus on the labels occurring rarely.

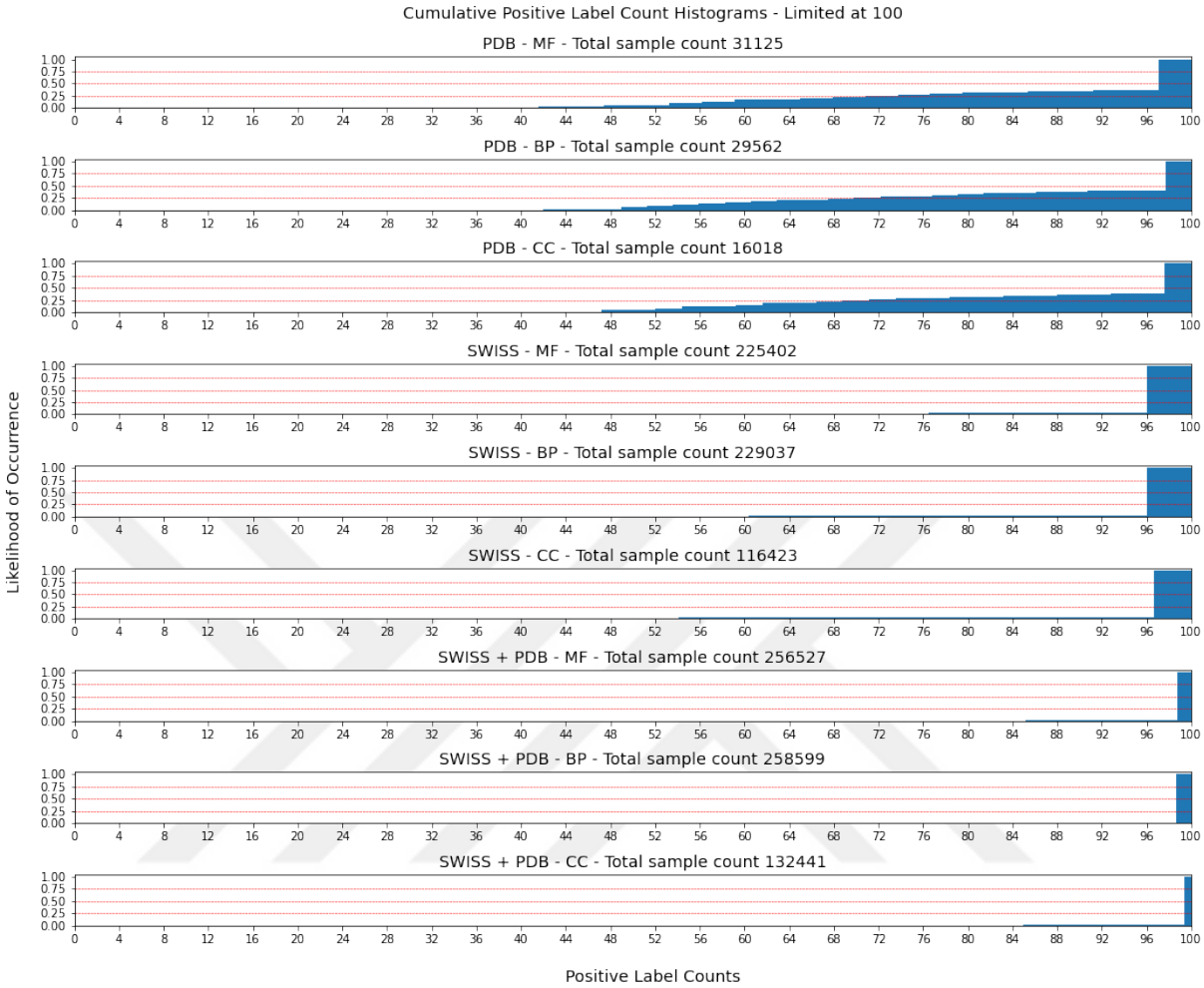

Figure A.4. The same information with Figure 3.2 but limited at 100 to focus on the labels occurring extremely rarely.

APPENDIX B: COMPARISONS OF PROT-BERT AND TEXT-GCN

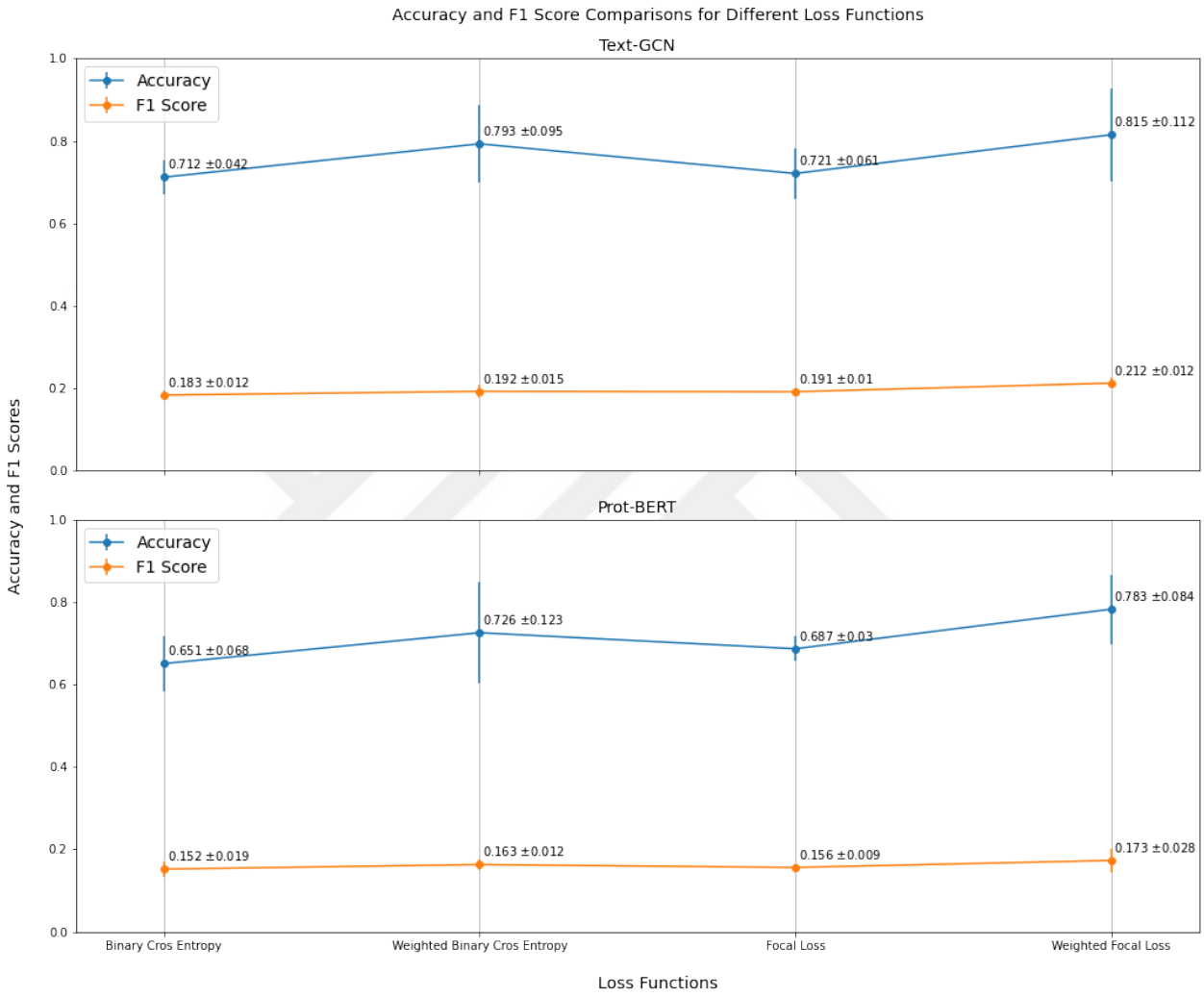

Figure B.1. Accuracy and F1 score comparison for different loss function options used during model trainings.

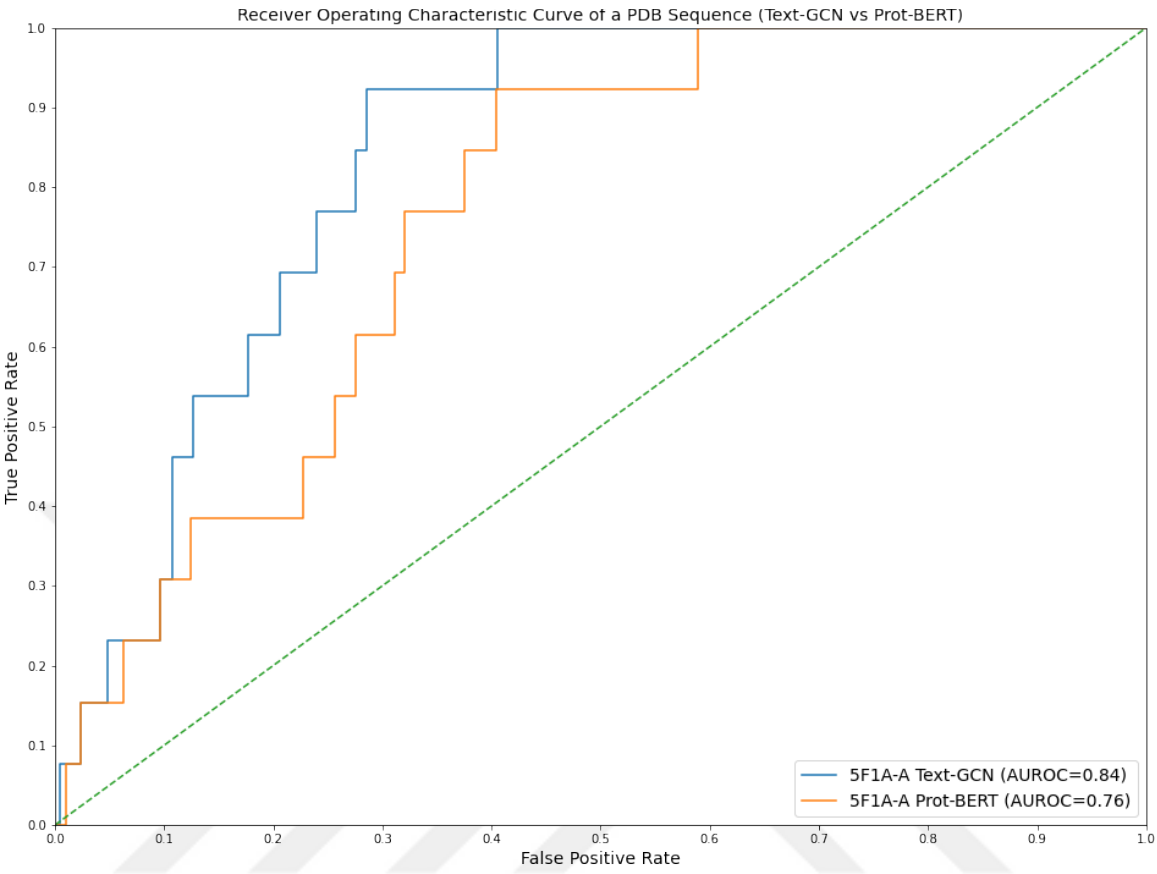

Figure B.2. Prot-BERT vs Text-GCN ROC curves and area under them for protein with PDB-ID *5F1A-A* where Text-GCN performed better than Prot-BERT.

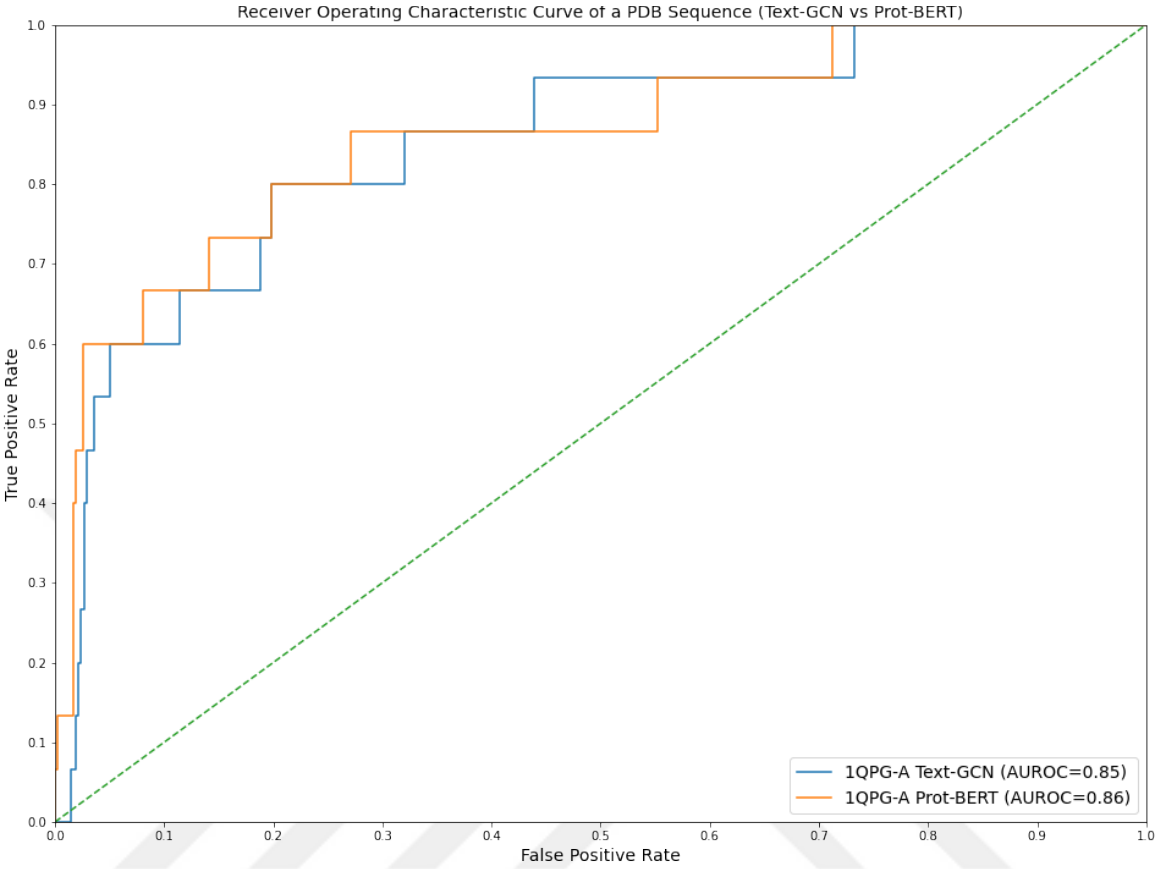

Figure B.3. Prot-BERT vs Text-GCN ROC curves and area under them for protein with PDB-ID *1QPG-A* where Text-GCN performed similarly to Prot-BERT.

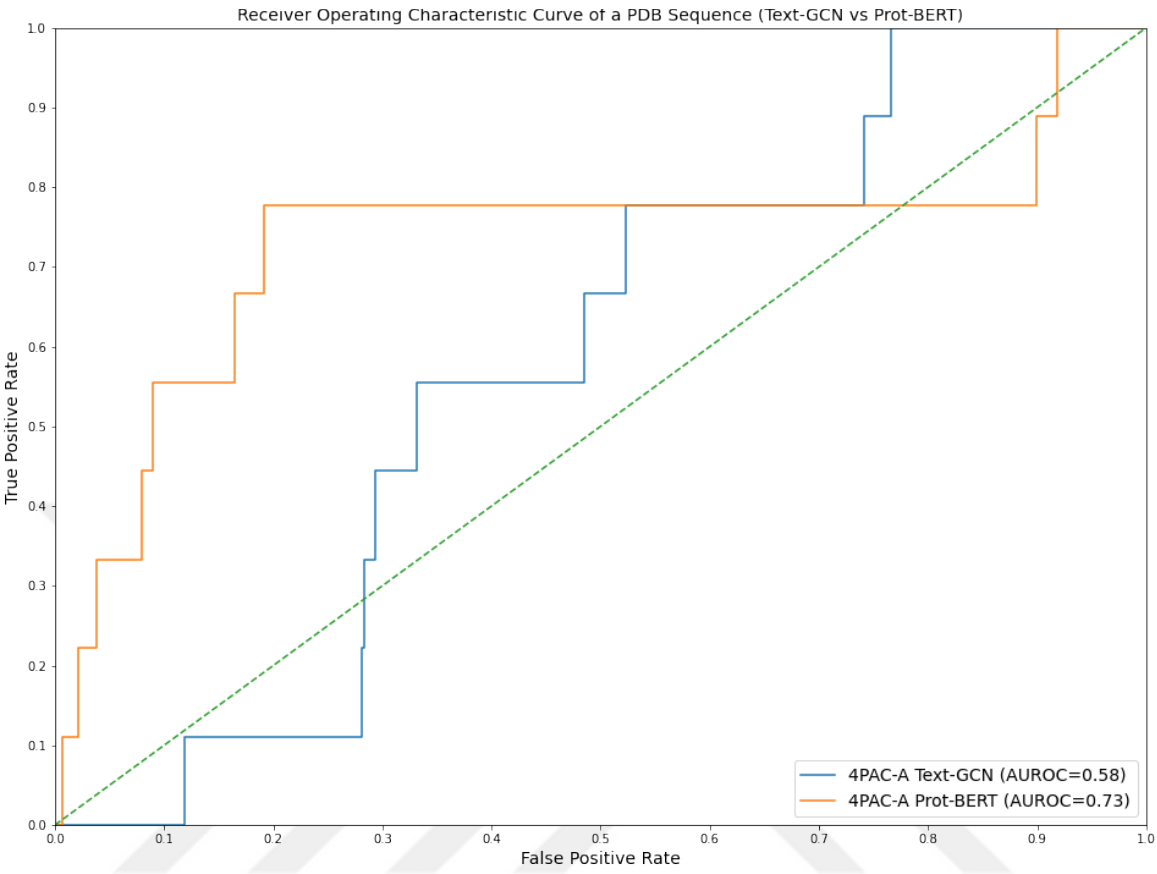

Figure B.4. Prot-BERT vs Text-GCN ROC curves and area under them for protein with PDB-ID *4PAC-A* where Prot-BERT performed better than Text-GCN which is slightly better than a random classifier.

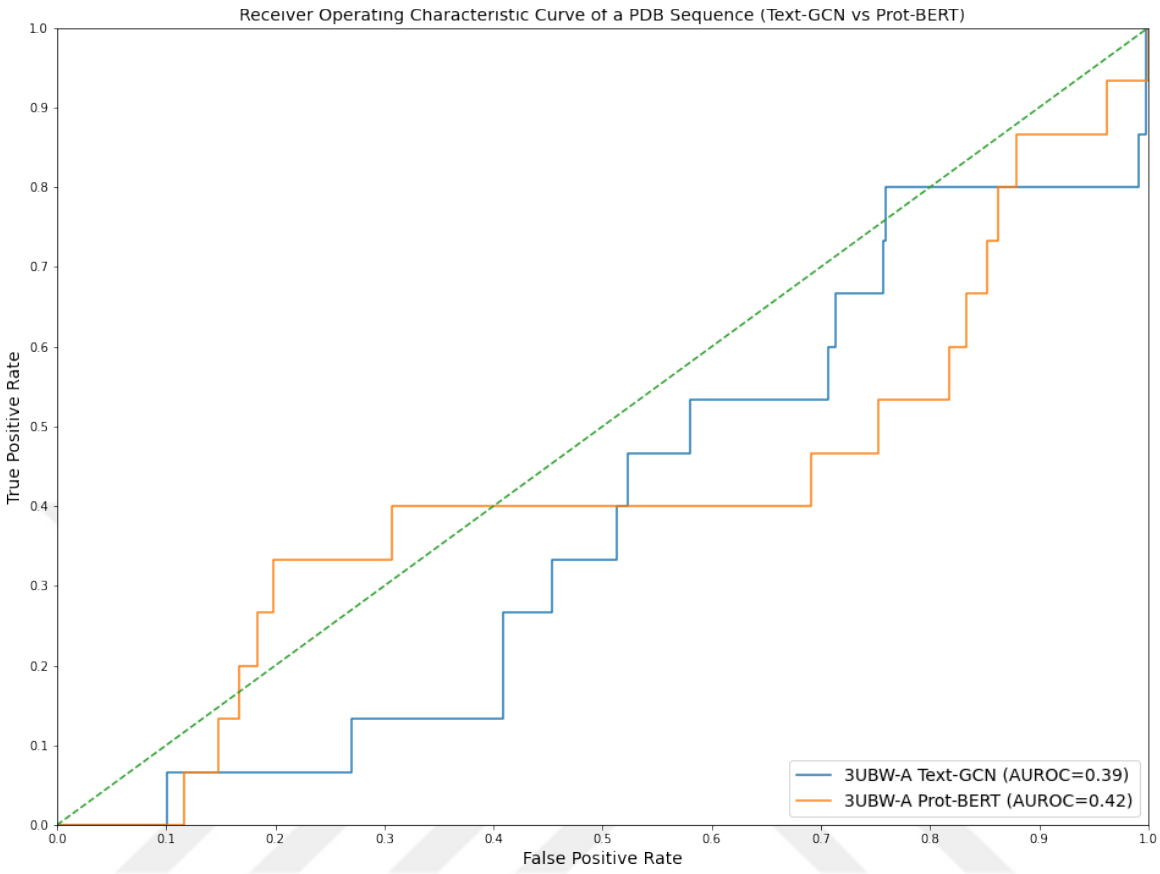

Figure B.5. Prot-BERT vs Text-GCN ROC curves and area under them for protein with PDB-ID *4PAC-A* where both models performed worse than a random classifier.

APPENDIX C: ADDITIONAL FIGURES OF PROT-BERT

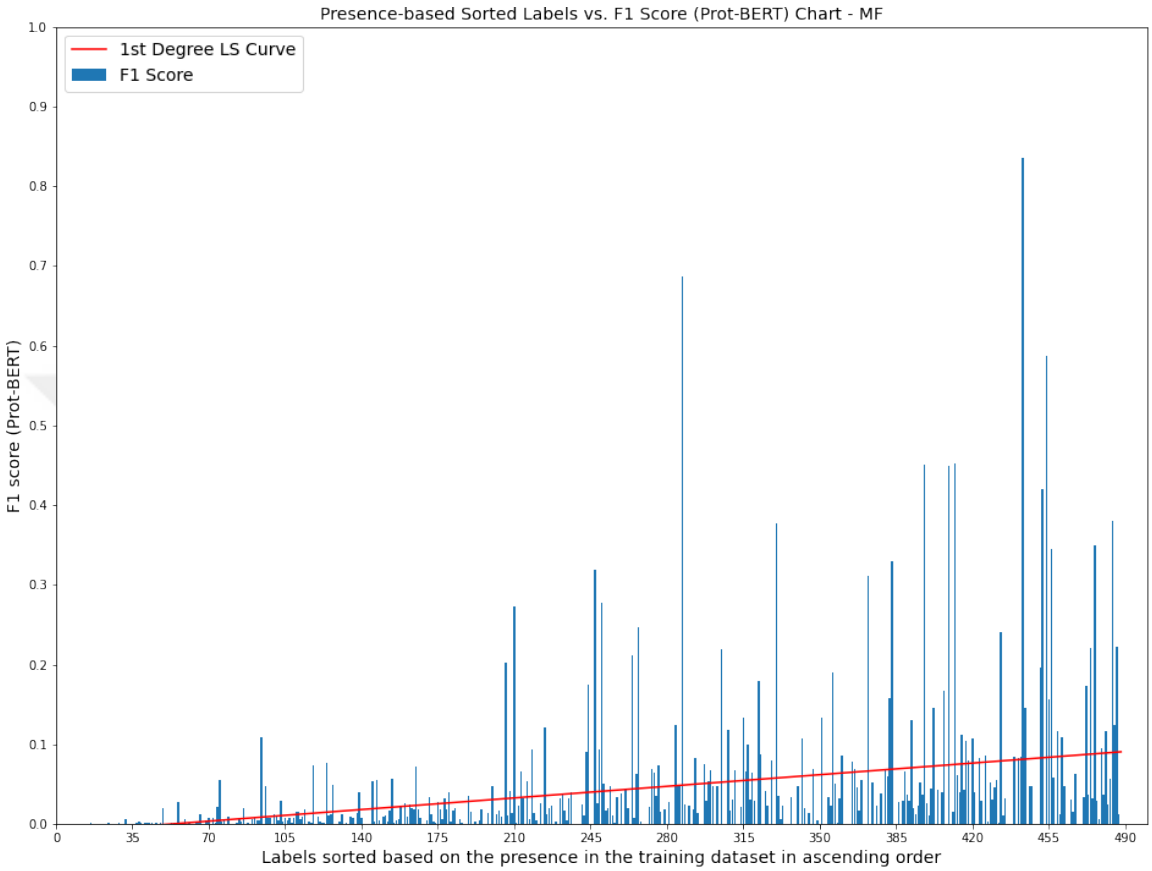

Figure C.1. Prot-BERT F1 scores of each label in MF ordered by their presence in ascending order.

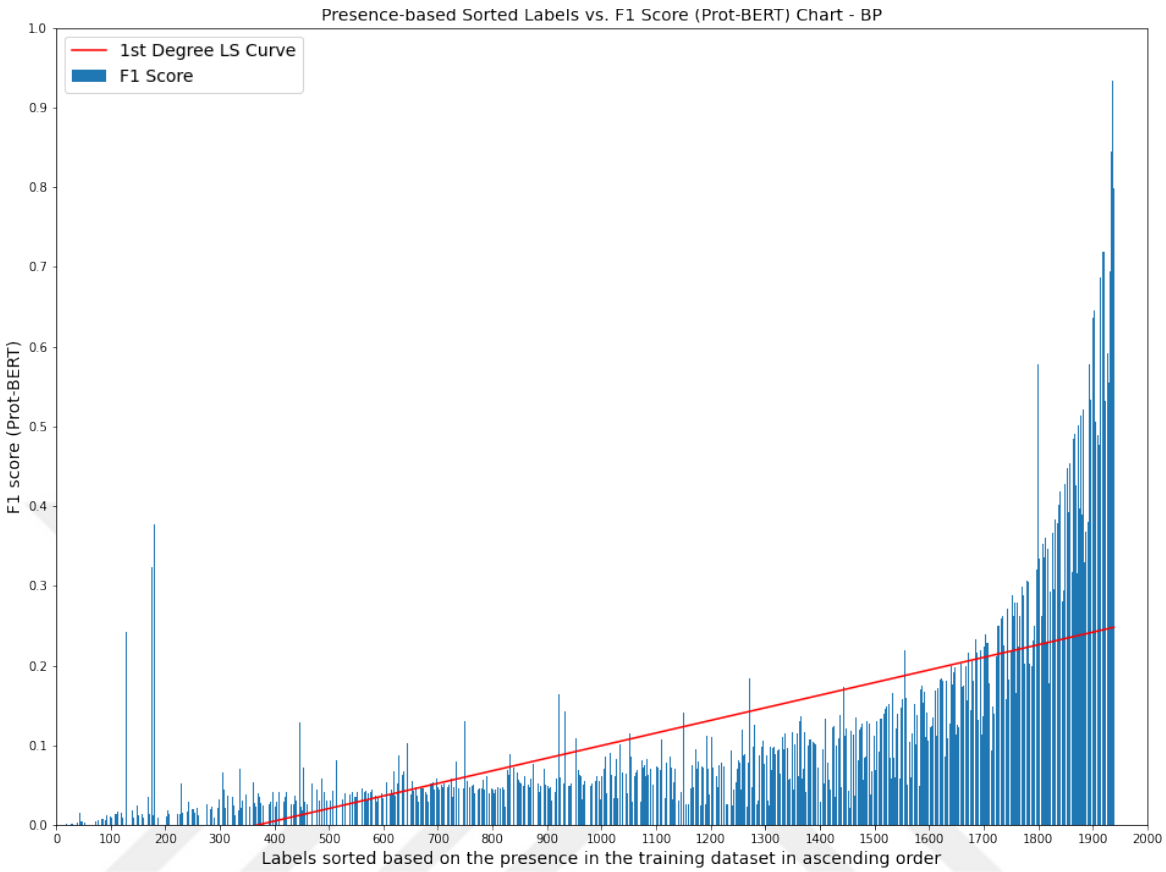

Figure C.2. Prot-BERT F1 scores of each label in BP ordered by their presence in ascending order.

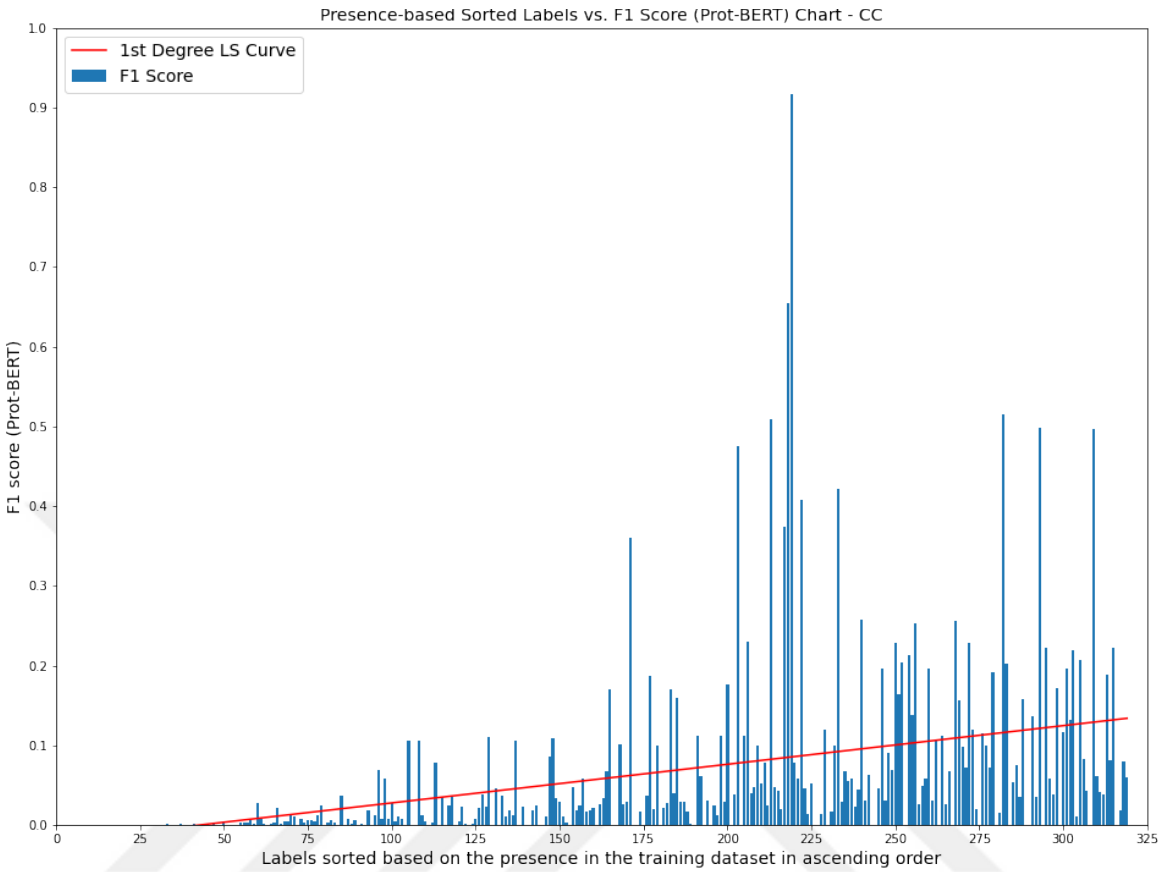

Figure C.3. Prot-BERT F1 scores of each label in CC ordered by their presence in ascending order.

APPENDIX D: ADDITIONAL FIGURES OF TEXT-GCN

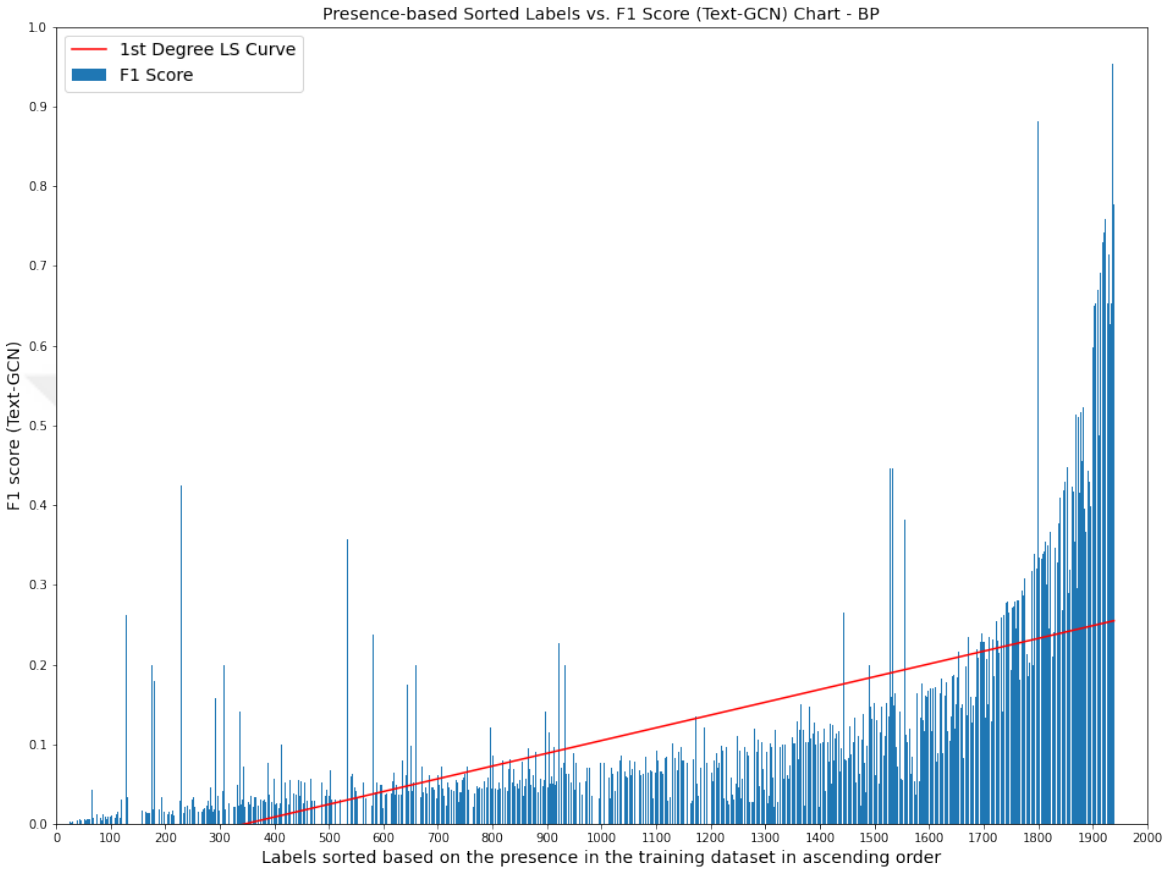

Figure D.1. Text-GCN F1 scores of each label in BP ordered by their presence in ascending order.

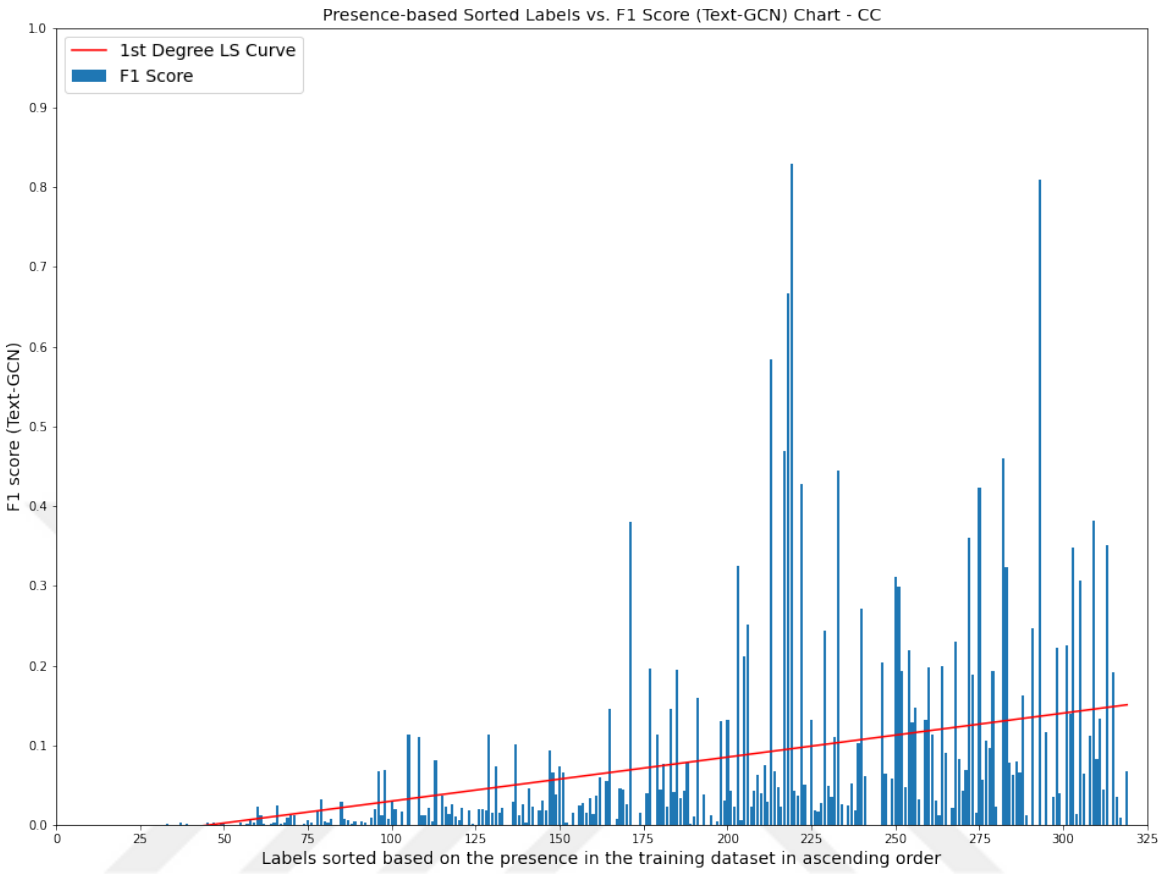

Figure D.2. Text-GCN F1 scores of each label in CC ordered by their presence in ascending order.