PREDICTION OF PATHOGEN-HOST INTERACTIONS WITH PROTEIN SEQUENCE EMBEDDINGS USING DEEP LEARNING

by

Büşra Oğuzoğlu

B.S., Computer Engineering, Bilkent University, 2019

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

2023

ACKNOWLEDGEMENTS

I want to thank Assoc. Prof. Arzucan Özgür for supervising this study and helping me to pursue my goals. It was a privilege to be able to work with her.

I thank Assist. Prof. Fatma Başak Aydemir and Assist. Prof. Öznur Taştan, for participating in my jury and providing important feedback.

I thank my parents, Öznur Oğuzoğlu and Hüseyin Oğuzoğlu, for always believing in me, being there when I need them, and supporting me unconditionally.

Finally, I want to express my gratitude to everyone that supported me in this journey, Cansu Damla Yılmaz and Gökçe Uludoğan for all their help, Pınar Süngü, and Sümeyye Ağaç for their support during my studies, Doğukan Nuhoğlu and Deniz Tuna for always listening to me and supporting me.

ABSTRACT

PREDICTION OF PATHOGEN-HOST INTERACTIONS WITH PROTEIN SEQUENCE EMBEDDINGS USING DEEP LEARNING

Infections caused by pathogens are a significant problem around the world. Determining protein interactions between pathogens and hosts is critical to understanding infection mechanisms and developing prevention and treatment strategies. Wet-lab experiments to identify protein interactions are expensive and time-consuming. Therefore, computational approaches have been proposed as a promising complementary solution. While 3D structures of proteins contain helpful information about protein functions, with advances in sequencing technology, 1D sequences of proteins are widely available and are often utilized because they are easier to process with less computational power. The main goal of this thesis is to develop a sequence-based approach for predicting pathogen-host protein interactions based on the hypothesis that protein sequences can be viewed as sentences, therefore, can be decomposed into chunks, which we refer to as protein words. We first adapt the Byte Pair Encoding (BPE) tokenization method from the field of natural language processing to protein sequences and then apply a graph-based approach using the Metapath2Vec algorithm to learn representations of sequences. The results show that incorporating a word-based representation of proteins improves the performance of the graph-based approach. In addition, two other methods for learning text representations, SeqVec and ProtBERT, are evaluated for predicting pathogen-host protein interactions. The results on three virus-host protein interaction datasets show that the sequence-based protein representation approaches are promising and achieve comparable performance to the state-of-the-art methods.

ÖZET

PATOJEN-KONAK ETKİLEŞİMLERİNİN DERİN ÖĞRENME YÖNTEMLERİ KULLANILARAK TAHMİN EDİLMESİ

Enfeksiyonlar, dünya çapında büyük bir sorundur. Konak ve patojenler arasındaki protein etkileşimlerini belirlemek, enfeksiyon mekanizmalarını anlamak, önleme ve tedavi stratejileri geliştirmek için kritiktir. Bu etkileşimleri belirlemek için kullanılan laboratuvar deneyleri pahalı ve zaman alıcıdır. Bu nedenle, bilgisayar tabanlı yaklaşımların geliştirilmesi zaman ve maddi masrafları azaltabilecek umut verici bir çözümdür. 3 boyutlu protein yapılarına dair veriler, protein fonksiyonları hakkında yararlı bilgiler içerirken, dizileme teknolojisindeki ilerlemelerle 1 boyutlu dizi verileri yaygın olarak mevcuttur ve daha az bilgisayar gücü kullanılarak işlenebilirler. Bu tezin ana amacı patojen-konak protein etkileşimlerini öngörmede sadece dizi tabanlı bir yaklaşım geliştirmektir. Protein dizilerinin cümle olarak görülebileceği, dolayısıyla parçalara ayrılabileceği hipotezine dayanarak, patojen-konak etkileşimlerini tahmin etmek için dizi tabanlı bir yaklaşım geliştirilmiştir. Byte Pair Encoding (BPE) tokenize etme yöntemi protein dizilerine uyarlanmış, Metapath2Vec algoritması kullanılarak dizilerin temsillerini öğrenmek için grafik tabanlı bir yaklaşım geliştirilmiştir. Sonuçlar, proteinlerin kelime tabanlı temsillerini kullanmanın grafik tabanlı yaklaşımın performansını arttırdığını göstermektedir. Ayrıca, metin temsilleme öğrenme yöntemleri SeqVec ve ProtBERT de değerlendirilmiş ve grafik methodu ile karşılaştırılmıştır. Üç farklı veri kümesinde elde edilen sonuçlar, geliştirilen yaklaşımın umut verici olduğunu ve mevcut ileri seviye yöntemlere benzer performans elde ettiğini göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$c_t(d)$	Number of documents where t occurs
$f_d(t)$	Number of times t occurs in d
$f_w(d)$	Number of terms in d
h_i	ID of human protein i in a graph node
$IDF(t, D)$	Inverse document frequency of a word t in a corpus D
$log(i)$	Logarithm function
$p(i)$	Marginal probability of a word i
$p(i, j)$	Joint probability of words i and j
$PMI(i, j)$	Point-wise mutual information of words i and j
$TF(t, d)$	Term frequency of a word t in a document d
v_i	ID of virus protein i in a graph node

LIST OF ACRONYMS/ABBREVIATIONS

3D	Three Dimensional
APID	Agile Protein Interactomes DataServer
AUC	Area Under Curve
BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte-Pair Encoding
CNN	Convolutional Neural Network
DIP	Database of Interacting Proteins
ELMo	Embeddings from Language Model
FN	False Negative
FP	False Positive
GCN	Graph Convolutional Network
GO	Gene Ontology
HIV	Human Immunodeficiency Virus
HPIDB	Pathogen–Host Interaction Database
IMEx	International Molecular Exchange Consortium
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
MCD	Multi-Scale Continuous and Discontinuous
MLD	Multi-Scale Local Feature Representation
NCBI	National Center for Biotechnology Information
NLP	Natural Language Processing
NPV	Negative Predictive Value
PATRIC	Pathosystems Resource Integration Center
PDB	Protein Data Bank
PHISTO	Pathogen–Host Interaction Search Tool
PMI	Point-wise Mutual Information
PPI	Protein-Protein Interaction
PPV	Positive Predictive Value

RCNN	Region-Based Convolutional Neural Networks
RNN	Reccurrent Neural Network
SVM	Support Vector Machine
SeqVec	Sequence to Vector
TF-IDF	Term Frequency-Inverse Document Frequency
TN	True Negative
TP	True Positive
UniProt	The Universal Protein Resource

1. INTRODUCTION

Biological research has been historically done in laboratories, with physical experiments involving long processes and high costs. As an area affecting everyone worldwide, it is crucial to work on making these processes faster and easier. Hence, with the advancements in the computational area, many studies were focused on biological data. An essential challenge regarding biological data is their interpretation. Since this data is generally complex and difficult to validate, automatizing the data representation step can help to solve problems such as predicting interactions between biological structures like proteins.

1.1. Pathogen-Host Interaction Prediction Problem

This study focuses on host–pathogen protein–protein interactions, an area where researchers face challenges associated with infections and drug design [1]. Even though the prediction of protein-protein interactions is already an established research topic, most of the previous work in this area relies on hand-crafted features to model the information of the concerned proteins. Even though this is a reliable way to create data, it is very time consuming and costly, considering the nature of the materials concerned. With the improvements in Machine Learning and Deep Learning, new tools have emerged to create and generate representations for data. Even though most previous works utilize hand crafted representations, with newer studies it is shown that using Deep Learning methodologies focused on utilizing different methods can be adequate to learn representations of the data and use these representations on downstream tasks.

This work focuses on the prediction of pathogen-host interactions via the prediction of protein-protein interactions. There are different ways to represent proteins; however, in the simplest form, they can be represented in a 1D format similar to sentences. Hence, some approaches commonly used in the NLP domain can also be applied

to this problem. In this work, protein sequences are viewed as sentences and are divided into words in a non-overlapping manner. These sentences and words are used to train representation models, and classification is done using a classifier model. In order to divide the sequences, referred to as sentences, into sub-sequences that carry meaning, referred to as words, different tokenization algorithms can be used. One example utilized in this work is Byte-Pair Encoding, a subword-based tokenization algorithm that divides sequences into subsequences. It is possible to use graph-based algorithms on textual data to create representations. Since graphs are good at modeling the interactions between entities, this approach has been used in Natural Language Processing, as well as other domains, by modeling the interactions between words, sentences, and documents. We used a similar approach with protein sequences that are divided into words and added additional information, such as sequence similarity, to extend the representation model further. An algorithm to learn representations from heterogeneous graphs called Metapath2Vec [2] has been used for this approach. Its performance is tested on different datasets and compared with the previous methods and two different representation learning methods, namely ProtBERT and SeqVec.

A popular approach in the NLP domain to create representations are language models such as BERT (Bidirectional Encoder Representations from Transformers) [3] and ELMo (Embeddings from Language Model) [4]. These models are trained using vast amounts of data and can be used as a tool for representation learning. These models have also been trained with protein sequences to create sequence representations that can be used in downstream tasks such as predicting protein-protein interactions. SeqVec [5], a protein sequence representation model based on ELMo and ProtBERT [6], a similar model based on BERT, is used in this work. Their performance is compared with the graph model on the pathogen-host interaction task using three different datasets: DeNovo, Ebola, and H1N1.

1.2. Contributions

As the main contribution of this thesis, a graph-based method is used for representation learning using only protein sequences by treating them as sentences and dividing them into sub-sequences, referred to as words. With different setups on the created graph, including words and similarities between sequences, the effectiveness of words is shown.

Other than the graph based model, two traditional representation learning models trained for protein sequences using UniRef data [7] are used in the pathogen-host interaction model, and their results are compared with each other, the graph-based method and other previous methods. Results showed the general effectiveness of both sequence-based representation learning algorithms and the graph-based algorithm as they perform similarly or better than previous methods that used the same structure. Moreover, comparing other representation methods with the graph-based algorithm showed that similar results can be achieved with less training data.

2. BACKGROUND

2.1. Biological Background:

2.1.1. Proteins

Proteins are complex molecular substances that perform tasks crucial for life in all living organisms. These large molecules are made of amino acids and have complex 3D structures. Twenty unique amino acids are part of proteins naturally, the basic structure of alpha amino acid is as given in the figure 2.1. Even though using the entire structure of a protein may give more information about its operating mechanism, both working with these structures and creating sources that contain them are computationally expensive. The amino acid sequence of a protein also gives an idea about its function since it is seen that proteins with similar functions are composed of similar amino acid sequences, which can be processed efficiently [8].

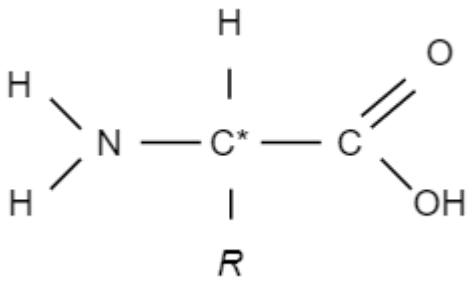

Figure 2.1. Basic structure of an alpha amino acid

2.1.2. Structure of Proteins

There are four levels of protein structure, primary, secondary, tertiary, and quaternary. The protein's primary structure is the most basic structure, consisting of a linear sequence of amino acids connected by peptide bonds. Secondary structure refers

to local connections within the single polypeptide sequence. Two general types of secondary structures are alpha helixes and beta sheets. The tertiary structure is the 3D construction of a single polypeptide chain. Lastly, the quaternary structure of the protein refers to interactions and structure between two polypeptide chains if the protein consists of more than one chain [8]. Even though higher-level structures may give a better idea about the function of a protein, since the sequence data is more commonly available and easier to work with, they are used in many studies.

2.1.3. Pathogen-Host Interactions

The interactions between pathogen and host are most commonly described as how and if a possibly illness-causing organism can sustain itself with the host organism. In this scenario, the host refers to a protein belonging to an organism that possibly experiences the illness when interacting with the protein belonging to a pathogen, most commonly a virus or a bacteria. The host can be human or other organisms depending on the problem. Even though this subject is mostly associated with "causing an illness" this is not always the case, as interaction may not occur. Therefore the problem can be formulated as a binary prediction problem, guessing whether the interaction occurs. As explained previously, proteins have different levels of structures, showing important characteristics. Because of this fact, structures of the proteins can be used to predict whether they interact or not and can be used in pathogen-host interaction prediction problems [9].

2.2. Databases

There are different databases created by gathering proteins and showing their structures and their interactions to be used in scientific studies. Some databases contain sequences, while others are structural databases or contain a specific type of interaction. Most commonly used databases include NCBI (National Center for Biotechnology Information) [10], UniProt [7] and PDB (Protein Data Bank) [11] Databases that were used in this work for different purposes are explained in the following sections.

2.2.1. UniProtKB/SWISS-Prot

UniProt Knowledgebase (UniProtKB) is a resource that contains manually and computationally curated data of proteins with information including name, description, amino acid sequence, taxonomic data, and citation information. If more data is available for the entrance, they are also added as extra information. The knowledge base contains two sections. One section contains manually annotated entries, and the other contains computationally analyzed and created data. These two sections are separated to disjoin manually annotated higher-quality data. They are referred to as "UniProtKB/Swiss-Prot"and "UniProtKB/TrEMBL". Swiss-Prot was created, manually annotated, and reviewed with experimentally proven results and conclusions. Later, the second section, TrEMBL, was created to use the data from genome projects and contains data that reviewers do not review manually. Both sections can be accessed from the UniProt website [7].

2.2.2. VirusMentha

Different data sources can be used to gather pathogen-host interactions. In this work, the primary dataset used in most experiments is called DeNovo, a dataset with train and test sections created by [12] from VirusMentha, which is a database consisting of virus-virus and virus-host protein interactions. It is an extension of a previous database called VirusMINT. Interactions on this database are manually reviewed, and they comply IMEx Consortium (International Molecular Exchange Consortium). This international work aims to provide protein interactions in a joint space. Also, since they aim to provide these interactions collectively, rules for the curation of new data have been developed, and reviewers done quality control for new data/articles to enter the system [13]. This effort provides reliability for the data that is available on VirusMentha. As mentioned, VirusMentha was created as an extension of a previous database called VirusMINT; however, entries from other existing datasets were also collected. These databases include IntAct, DIP, MatrixDB, and BioGRID. All data is available to view and download from the official website.

2.2.3. APID/IntAct

Other datasets used in this thesis are Ebola and H1N1, curated using APID and IntAct databases in Zhou et al.'s work on pathogen-host interactions [14]. IntAct is an open-source database for molecular interactions [15]. Similar to VirusMentha, interactions on IntAct comply IMEx Consortium. It provides both textual and graphical representations of molecular interactions. The database website provides an interface for search, and different identifiers, including Swiss-Prot, can be used. All data can be viewed and downloaded from the official website. APID is a similar database that includes experimentally proven protein-protein interactions and also has a website with a search tool and available data to download.

2.3. Technical Background

2.3.1. Byte-Pair Encoding (BPE)

Tokenization is a prevalent task in NLP, involving dividing sentences or documents into small pieces referred to as tokens. These tokens are usually words in the sentence or can be smaller parts of the words and word groups that have significance together. Different methods can be used for this task ranging from using space to divide sentences to training complex models. Byte-pair encoding is one of the methods commonly used for tokenization tasks. It focuses on dealing with rare or unknown words during the process by dividing the sequence or sentence into subwords. This method was first introduced in the paper 'Neural Machine Translation of Rare Words with Subword Units' [16]. Generally, the idea behind this method is to treat encountered rare or unknown words as a sequence of known sub-words.

The main inspiration for this method is an algorithm for lossless data compression. This algorithm replaces the most common pairs of bytes in a string with a replacement that originally does not appear on that given string. As an example, lets say our starting string is 'dbcbbdbcba', steps can be seen on figure 2.2. As the first step,

vocabulary is created and initialized. After that, each word is represented as a sequence of characters, and a unique end token is added. After this, pairs of characters are counted for all items in the vocabulary.

dbcbb**db**cba (Replace db with X)
Xcbb**Xcb**a (Replace cb with Y)
XYb**XY**a (Replace XY with Z)
Zb**Z**a (Compressed string)

Figure 2.2. Steps for BPE algorithm

Starting from the most frequent pair, similar to the compression example, all occurrences of these characters are merged together (instead of replacing them with another character as in the example). This operation continues until a finishing criterion is reached.

Table 2.1. Sample traceback table.

Byte Pair	Replacement
Z	XY
X	db
Y	cb

This criterion can be set as the vocabulary size or the number of operations; therefore, the algorithm may stop when the vocabulary size becomes a set value or the number of operations exceeds another set value to prevent the algorithm from running indefinitely. Since we represented protein sequences in a similar form to sentences, we can use this method on them. We can assume amino acids represent letters, and a group of amino acids can be thought of as words in a sentence or a document. Since words are made of letters that appear together frequently in the natural language, using our analogy, sequences can be divided into meaningful tokens or “words“ using the BPE algorithm. Later these words can be connected to their sequences, analysis

can be done regarding which words appear more frequently, and they can be used in a graph setting and for representation learning purposes.

2.3.2. Embeddings from Language Model (ELMo)

In this thesis, the SeqVec (Sequence to Vector) model based on ELMo (Embeddings from Language Model) [4] is used in comparison with other representation learning methods. ELMo is a commonly used language model utilized in the NLP domain for representation learning. For its time, ELMo was revolutionary. Before ELMo, methods for word embeddings used information such as frequency, but they did not consider the context of the sentence that the word belonged to because of their model structure. In a sentence, the context matters and gives the words within that sentence their meanings; therefore, a method that can use this information would give better results. ELMo consists of bidirectional LSTMs. Using bidirectional LSTMs causes the model to consider the words that come after a specific word, previous ones, and the whole sentence. These networks are commonly used with long sequences since they are built for learning long-term dependencies within the given sequence.

With the popularity of language models like the ELMo model in the NLP domain, these models were later trained with protein sequences and can be used. An ELMo model trained on UniRef50, named SeqVec, is publicly available and can be used as a representation learning tool for protein sequences [5].

2.3.3. Bidirectional Encoder Representations from Transformers (BERT)

BERT (Bidirectional Encoder Representations from Transformers) is a deep learning model based on transformers. It is used commonly in natural language processing tasks since it can achieve state-of-the-art results in important tasks such as Question Answering, Named Entity Recognition, Sentiment Analysis, Natural Language Inference, and others. Transformers are a model architecture consisting of two parts, an encoder, and a decoder. They use the attention mechanism, which means the model

can identify the importance of inputs or parts of the input and give variable weight to different parts. This ability makes this architecture different from recurrent models used commonly and gives excellent results in NLP-related tasks.

The original BERT model was trained on an extensive dataset, namely Wikipedia and Google BookCorpus data, and training took four days with 64 TPUs (Tensor Processing Units). Later, smaller BERT models such as BERT-Mini, BERT-Tiny, DistilBERT, and others were introduced. While these models can run faster, they can still give excellent performance [3].

With the BERT model getting very popular in the NLP domain, it was also trained for protein sequences, the model being called ProtBERT [6]. The model can be used directly from the HuggingFace Transformers library [17]. Training is done using only protein sequences. Every sequence is treated as a document instead of a sentence during the training process. This approach is different from the original BERT's training process, as it also uses the next sentence prediction objective, which is not used in ProtBERT. Despite the differences with the original, the results showed that the ProtBERT model could capture essential properties of the protein sequences. This model has also been tested and compared with the ELMo based SeqVec model in this study.

2.3.4. Metapath2Vec

Metapath2vec is a method for representation learning from graphs with multiple types of nodes and edges (heterogeneous graphs) [2]. The network embedding problem has been studied in previous works like DeepWalk [18], and Node2Vec [19]. In these works, it was argued that information networks could behave similarly to natural languages. In these networks, some nodes can appear together because they have a common interest together or have meaning together. An example of this can be viewing a network with friend groups and mapping those groups based on their interactions, relationships, and the way they choose to be connected as a social language. This

construction is similar to words appearing together frequently in sentences and having a mutual meaning. Following this analogy, the methods we use in the NLP domain to learn representation, such as Word2Vec [20], can also be used for this scenario.

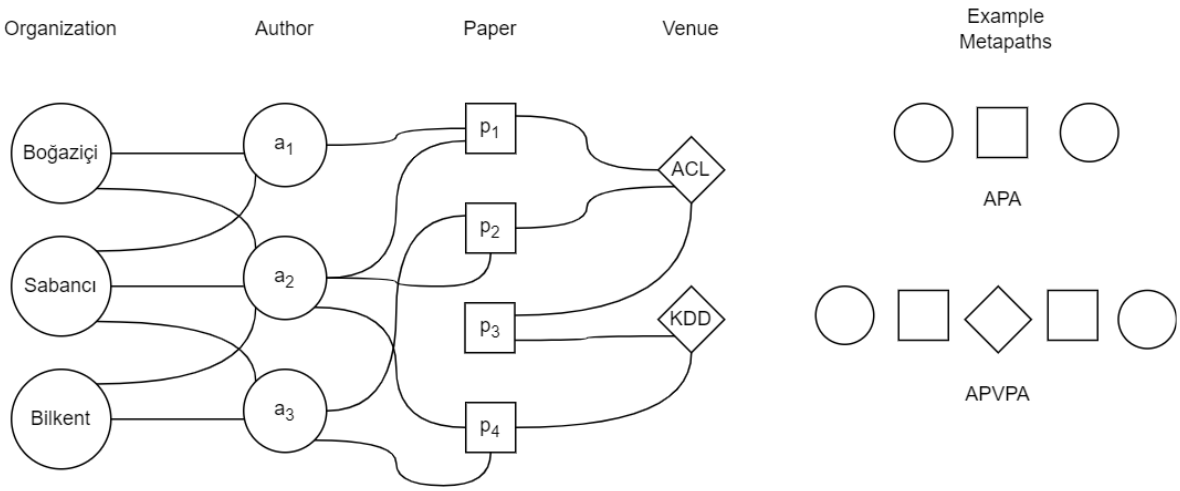

Figure 2.3. Example heterogenous graph and associated metapaths.

The biggest downside of these studies was that they work with homogenous graphs. In the real world, information networks usually contain more than one type of node and edge. To tackle this problem, these studies were extended to heterogeneous graphs with Metapath2vec. In the example in figure 2.3, we can see some example metapaths that can be used with the given graph. Using these, we can learn node embeddings with a methods such as Word2Vec [20].

2.3.5. Point-wise Mutual Information (PMI)

PMI is a measurement that shows how correlated two points (words, in this case) are. Calculation is done for words w_1 and w_2 as

$$PMI(w_1, w_2) = log\left(\frac{p(w_1, w_2)}{p(w_1)p(w_2)}\right) \tag{2.1}$$

where $p(w_1, w_2)$ refers to the joint probability (the words occurring together), and $p(w_1)$ and $p(w_2)$ are their probabilities separately. From this calculation, if two words have a high PMI score, it means that they usually occur together in sentences. If two words occur together frequently, they have a high chance of having a meaning together.

During the experiments with the graphs, the PMI score is used to determine if two sequence pieces (protein words) are correlated enough to add an edge between them. Using a filtering score for this is essential because adding too many edges would make the graph very sparse, making it hard to get the critical information.

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

TF-IDF is an algorithm used commonly in the Machine Learning and Deep Learning domain and statistics to determine how important a term is for a given document within a set of documents [21].

TF-IDF consists of two components: term frequency and inverse document frequency. Term frequency referred to as $TF(t, d)$ is calculated as

$$TF(t, d) = \frac{f_d(t)}{f_w(d)} \tag{2.2}$$

where $f_d(t)$ shows number of times t occurs in d, and $f_w(d)$ shows the total number of terms in d. The equation shows how much/frequent the word/term t occurs in the document d. Document frequency shows how much the word/term appears in all documents. Inverse document frequency is referred as $IDF(t, D)$ calculated as

$$IDF(t, D) = \frac{|D|}{c_t(d)} \tag{2.3}$$

where $|D|$ shows the total number of documents in corpus D, and $c_t(d)$ shows the number of documents where t occurs. The calculation shows the inverse correlation between a word and all documents, referred as D. Calculation of $TF - IDF$ value is the combination of these two values:

$$TFIDF = TF \times IDF \tag{2.4}$$

and a high TF-IDF score shows that said word is relevant for the document that we are interested in since, compared to its prevalence in all documents, it appears more in the said document. Similar to PMI score, for the graph experiments, the TF-IDF score is used to filter out the edges between sequences and words that may not hold much information to prevent the graph from being too sparse.

2.3.7. Needleman-Wunsch Algorithm

The Needleman-Wunsch algorithm (also called global alignment or optimal matching technique) is used in bioinformatics to determine the similarity between two proteins or nucleotide sequences by aligning them. The algorithm aims to find the least costly alignment possible between two sequences. This is done by calculating the possible alignments using the dynamic programming approach and giving scores to them based on a scoring system.

As an overview, we can follow an example. We can have two sequences to compare, ACGATC and ACACTC. These sequences can be aligned in many different ways; however a possible alignment can be made on these sequences by making a deletion of third character and left a mismatch for forth. There can be other ways to align these two sequences. A scoring system is used to evaluate how good a possible alignment is. Even though different ways can be followed for scoring as well, a basic but useful scoring system can be constructed as follows:

- If letters match (they are in same place), give $+1$
- If letters match by a gap (can be matched after deletion or insertion), give -1
- If letters do not match, give -1

following this scoring system, our alignment gets the score $++--++$ (match, match, deletion, mismatch, match, match) and it equals 2.

In order to find the least costly alignment, we need to consider all possible alignments and score them. Even though this is a relatively easy task for a small sequence, as the example, it is challenging for actual protein sequences. In order to tackle this problem, the dynamic programming approach is used. The main idea behind this approach is to find the best alignments of small subsequences of a larger sequence and use them to find the global optimal alignment.

An overview of the algorithm is as follows, first, a scoring and a traceback matrix is created. After that, these matrices are filled using a scoring scheme, and traceback matrix is also filled during this process. At the end, best possible alignment is found using the traceback matrix.

Table 2.2. Sample scoring table.

		A	C	G	A	T	C
		-1	-2	-3	-4	-5	-6
A	-1	1	0	-1	-2	-3	-4
C	-2	0	2	1	0	-1	-2
A	-3	-1	1	1	2	1	0
C	-4	-2	0	0	1	1	2
T	-5	-3	-1	-1	0	2	1
C	-6	-4	-2	-2	-1	1	3

Table 2.3. Sample traceback table.

		A	C	G	A	T	C
		←	←	←	←	←	←
A	↑	↖	←	←	←	↖	←
C	↑	↑	↖	←	←	←	↖
A	↑	↖	↑	↖	↖	←	←
C	↑	↑	↖	↖	↑	↖	↖
T	↑	↑	↑	↖	↑	↖	←
C	↑	↑	↖	↖	↑	↑	↖

According to matrices that were created on 2.2 and 2.3 according to our example, the optimal score corresponds to 3, which is the cell in the bottom right corner, and

by tracing back the table, we can find the route that gives the best score. The route that gives the best score shows us the best possible alignment.

Table 2.4. Sample traceback table.

		A	C	G	A	T	C
		←	←	←	←	←	←
A	↑	↖	←	←	←	↖	←
C	↑	↑	↖	←	←	←	↖
A	↑	↖	↑	↖	↖	←	←
C	↑	↑	↖	↖	↑	↖	↖
T	↑	↑	↑	↖	↑	↖	←
C	↑	↑	↖	↖	↑	↑	↖

Following the calculations, alignment of the sequences are CA-CATA/CAGC-TA, and this is the best possible alignment according to algorithm. This algorithm is used in the graph setting to give similarity scores to sequence pairings. The score between the same type of sequences is used to create edges between them. These scores were also used to create embeddings for test and validation sets. Details of the calculation and usage in the graph are explained in the following sections.

3. RELATED WORKS

Many works on pathogen-host interactions integrate different sets of data on their pipeline. Some of these works focus on utilizing different features from available data, such as gene ontologies, to improve their results. Others focus only on sequence information similar to this work. While most of these works depend on information related to proteins, such as sequence information, protein structure, and their known interactions, some others integrate other available data like clinical symptoms [22].

An example of the previous approaches integrating different data into their model is hivPPI [23]. In hivPPI, authors have used gene ontology (GO) [24] similarity features, information regarding topological properties of human sequences, similarity information regarding sequences belonging to HIV and other known binding partners of the human proteins and features regarding functional motifs of the proteins. These features were used in a supervised learning framework to predict the interactions between HIV and human proteins as interacting and non-interacting. Another example is DeepViral [25] The authors also integrate clinical symptoms in this work, extending their sequence-based model. They first embed human proteins and pathogens (specifically viruses in their case) based on their functions, taxonomic classifications, and with extra background knowledge from biological ontologies. They claimed that extending their model with phenotype features improved their results compared to their previous work, which only used sequence information. They have used the DL2Vec method to generate the feature embeddings, a graph-based machine learning method to predict genes from phenotypes [26]. As for the data, the HPIDB database was mainly used, which is a database containing pathogen-host protein-protein interactions [27]. The authors also used the dataset used in DeNovo [12], which is a curation from VirusMentha [28] as a supplementary dataset to compare their results with previous methods, like DeNovo [12], and RCNN model [29]. Both datasets are easily accessible, the DeNovo dataset being smaller and possible to work efficiently with hardware limitations. As a third example, S-VGAE, a representation learning method, tries to utilize data

from protein networks, such as degree, position, and neighbor information, on top of sequence information [30]. Methods based on networks have been used in other tasks, including protein sequences, giving promising results, and also has been used in this thesis.

A general problem that is mentioned by many papers is the sparsity of available data. Even though there are some databases and tools for PPI like PATRIC [31], PHISTO [32], VirHostNet [33], HPIDB [27], and VirusMentha; [28], these databases only represent a small number of Host-Pathogen PPI's [1]. This has led researchers to search for better ways to utilize the available data; therefore, there are different choices regarding using these tools. While some works choose supervised approaches, some of the methods try to use semi-supervised or unsupervised learning methods for this task since the available data is highly imbalanced, negative data being unavailable and created mostly using random sampling [34] [1]. Some other works try to solve this problem using data augmentation and random projection [35].

Following the problem with the data sparsity, while some of the studies purely depend on the sequence information, some try to represent this information in a more informative, different manner to extract more information. Methods like AC (auto covariance), its variants like ACC (autocross covariance) [36] [37], MCD (multi-scale continuous and discontinuous) which can give information about sequentially distant amino acid combinations [38] or MLD (multi-scale local feature representation) which can capture information from segments of different length [39] are used for feature extraction. More recent studies usually use sequence-embedding-based machine learning and deep learning methodologies. Different combinations of embeddings and classifiers are utilized in different works. Among these, some use machine learning based approaches for classification, such as SVM [37] or random forest [39]; others use deep learning based methods. For example, one of the models created by Yang et al. [40] uses Doc2vec [41], sequence embedding in an unsupervised fashion, to capture the information and random forest for binary classification, to classify pairs from pathogen and host as interacting and non-interacting.

Another approach is to view protein sequences as sentences and to view this problem similarly to an NLP task. In a different study, DeepSequencePPI [42], as the first step authors have 'tokenized' the sequences, and they have modeled them as '3-grams'. Then they used a deep neural network with embedding and recurrent layers. They have compared their results with similar works like DeepPPI [43], which follows a similar idea but does not use recurrent networks, but instead uses a fully connected feed-forward neural network. Another work, DNN-PPI [44], follows a different approach by combining CNN and LSTMs. In this work, they treated amino acids as words and protein sequences as documents inspired by word2vec. Also, LSTM was chosen instead of regular RNN because of its advantage in exploding and vanishing gradients problem. They give interacting sequences as pairs to the CNN-LSTM network, then concatenate the two outputs from the pair and feed that to a feed-forward neural network, which will make the final classification.

Also, following this kind of studies, some methods specifically for biological sequence embeddings have emerged. Some of these methods are seq2vec, a Euclidean space representation for biological sequences [45] and SeqVec (Sequence to Vector), which is based on NLP model ELMo to model protein sequences as embeddings with transfer learning [5]. This work tested the SeqVec model on DeNovo, Ebola, and H1N1 datasets. These models' authors also tested them on different tasks like subcellular localization prediction. With the BERT model getting very popular in the NLP domain, it was also trained for protein sequences, the model being called ProtBERT [6]. This model has also been tested and compared with DeepViral and SeqVec, as well as the graph-based method that is introduced in this study.

4. MATERIALS AND METHODS

4.1. Datasets

4.1.1. DeNovo - VirusMentha

In order to be able to compare the new methods with existing ones, datasets that were created and used by previous works were considered. The first one of these datasets is DeNovo's virus-human protein interaction dataset. It was created by Fatma et al. [12]. They have prepared this set by collecting interactions between human-virus proteins from the VirusMentha database (accessed in June 2014) [28]. They have gathered a total of 2.357 human proteins, 453 viral proteins, and 5.753 interactions. The virus proteins in the set belong to 173 different species and 25 different subfamilies. In order to create the negative subset, they have followed an approach that was followed by previous works like [46] by randomly sampling proteins that do not appear in the positive set. This approach has some downsides since knowing if the interaction exists is different from knowing there is no interaction; however, many studies have followed this approach due to the unavailability of negative data. After preparing the sets, sequences were gathered from UniProt [12].

In this work, dataset was initially gathered using utility functions from DeepViral code to be able to reproduce their results and use the same partitions as them in the experiments. The data did not contain the sequence information, and sequences were gathered from the UniProtKB. After gathering the sequences, 9230 data points were available in the train portion of the dataset, and 840 were available in the test set. These are the ones that have sequence data available for both sides, human and virus.

The authors of DeepViral used 9.754 (5020 positives and 4734 negatives) data points in their experiment for training. For the test, 850 data points are available in DeepViral, 425 negative and 425 positive. This difference is caused by the fact that

some data points became obsolete after the authors of DeepViral created their dataset. To be able to make comparisons, the obsolete data were also gathered by hand and added to the set. This way, the created dataset became identical to DeepViral's DeNovo partition, and we could compare the results.

Table 4.1. DeNovo dataset statistics.

	Train	Test
Positive Samples	5020	425
Negative Samples	4753	425

The DeNovo paper was published in 2016, and it is seen that some data become obsolete on UniProtKB over time. The history of changes can be seen on UniParc, and the sequences used before can be reached from there. Unfortunately, this process could not be automated. The list of UniprotKB of sequences that become obsolete for train and test sets are reported as follows:

- Train Set Obsoletes: P04544, P63233, P63231, Q85737, P08107, P0CG06, P62158, P03074, Q13748, Q17R26, Q5SVS2, Q7Z469, Q9BXM8, Q5SVS2.
- Test Set Obsoletes: Q9BXM8, P03076, P08107, P62158

as it would be meaningful to update the partitions with up-to-date data. To test the data representation methods SeqVec and ProtBERT, sequences had to be cut to a specific length as a preprocessing step. This process is followed to avoid memory issues, which is a problem with long sequences, especially with the ProtBERT model. Since the variability of sequence length has an effect on results, an analysis of used datasets has been done to show how variable the length of sequences is. Results for the DeNovo dataset have been shown in 4.2.

Table 4.2. DeNovo dataset length statistics.

Length	**Human**	**Virus**
>1000	433	42
>750 <1000	266	26
>500 <750	550	68
>250 <500	902	118
>0 <250	612	165

Another critical statistic regarding the dataset is how similar the sequences in the test portion are compared to the train portion. Since the aim is to simulate behaviors of the models in the case of novel viruses, having a test set with non-similar viruses with the train set would be ideal. However, making them completely random could also be unrealistic since new viruses can result from mutations. With these in mind, it is crucial to assess the similarity between portions of the set. The similarity between virus sequences belonging to test and train portions was calculated using the Needleman Wunsch algorithm, and statistics are given on the table below. This table shows the number of unique virus sequences in the test set and the similarity ranges. These are calculated by number of viruses on that similarity range when compared to viruses on the train set. For example for a virus sequence in the test set, if any of the train sequences have similarity value more than 90, that is added to the $> 90, < 100$ section. For Denovo dataset, these statistics show that most of the viruses are less than %25 similar to train set, showing the test portion is quite different than the train portion of the dataset.

Table 4.3. DeNovo dataset similarity statistics.

Similarity Value	**Sequence Count**
>0 <25	26
>25 <50	6
>50 <75	4
>75 <90	0
>90 <100	2

For validation, two different sets were created. The first one was created by sampling 0.1 of the training dataset directly, following the approach of DeepViral. This approach causes some of the virus sequences to be the very similar in the train and validation dataset (we have seen on the previous table that over %90 range is populated in this dataset), so a second approach has been used to cross-validate the results and analyze the graph model. The second validation set has been created by further sampling the dataset, so that train and validation datasets do not have any common virus sequences. This has been done by randomly selecting ten different viruses from the training dataset and removing their related data (interactions containing these selected viruses) from the training set. Experiments were run on both to decide the default hyperparameters for different models.

4.1.2. Zhou et al's Ebola and H1N1 Datasets

The second set of data that was gathered for this work is Zhou et al.'s Ebola and H1N1 Datasets. To create these datasets, Zhou et al. gathered human-virus pathogen-host interactions from the databases APID (Agile Protein Interactomes DataServer), IntAct (IntAct Molecular Interaction Database), and Mentha (The Interactome Browser). There are 12,157 interactions (containing 332 viruses and 29 hosts, including human and other animals) collected from these three sources. Even though hosts other than human are available, their portion in this data was minuscule, and they were not included in the sets later. Sequences were gathered from UniProt.

After that, they created two datasets for training and testing using this data: H1N1 and Ebola. The train portion of the Ebola dataset contains interactions between humans and all included viruses, excluding the Ebola virus. Similarly, the train portion of the H1N1 set contains all of the interactions except the H1N1 virus. The test set contains interactions of Ebola and H1N1 viruses, respectively. This decision is to simulate how the model behaves when a new virus comes out and what if it was Ebola or H1N1 virus at the time. To create the negative samples, they used a similar negative sampling approach; however, they removed sequences with a similarity higher than 0.8 to positive samples calculated by CD-HIT-2D [14]. This is an automated comparison program for protein sequences using a clustering algorithm called CD-HIT. CD-HIT-2D version compares two datasets and finds similar protein sequences in the second dataset to the first one. Using this method, they created the same number of negative samples as positive ones. Statistics of both sets are given below:

Table 4.4. Ebola dataset statistics.

	Train	Test
Positive Samples	11,341	150
Negative Samples	11,341	150

Table 4.5. H1N1 dataset statistics.

	Train	Test
Positive Samples	10955	381
Negative Samples	10955	381

Statistics of the both sets based on sequence lengths are given below. We can see that similarly to DeNovo, most sequences on Ebola and H1N1 datasets are on the 250-500 range. However, there are many sequences that are larger than 1000 characters as well.

Table 4.6. Ebola dataset length statistics.

Length	**Human**	**Virus**
>1000	1106	70
>750 <1000	761	39
>500 <750	1561	84
>250 <500	2818	191
>0 <250	1834	247

Table 4.7. H1N1 dataset length statistics.

Length	**Human**	**Virus**
>1000	1103	70
>750 <1000	783	37
>500 <750	1607	84
>250 <500	2884	190
>0 <250	1847	240

Unlike DeNovo, these two datasets have been arranged to simulate a situation where a single new virus emerges, namely H1N1 and Ebola. Therefore their test sets only contain these viruses and their variants (three unique sequences total in Ebola, 12 sequences total in H1N1). Because of this reason, for all unique sequences, similarity statistics are on the 0% to 25% range except for one sequence in H1N1 set and one sequence in Ebola set which are on 25% to 50% range.

For this work, in order to get the same portions of the datasets, they were gathered from another work that has used them and achieved excellent results; namely, MotifTransformers [47]. Train and test portions of both sets can be gathered from their GitHub page, as they are shared publicly for replication.

4.2. Models

As for the premise of this thesis, different representation models and methods for protein sequences were tested in the context of the pathogen-host interaction prediction problem. These models include representation learning models ProtBERT [6] and

SeqVec [5], which are BERT and ELMo architecture-based pretrained protein sequence representation models. Also, a model based on the idea of dividing protein sequences into sub-pieces, or words, a graph-based model using the Byte-Pair Encoding tokenization algorithm, was introduced. Details of these models are explained in the following sections.

4.2.1. Data Representation Models

4.2.1.1. ProtBERT. Pre-trained ProtBERT model was gathered from HuggingFace used for feature extraction. Since this model requires high computational power, it could not be trained from scratch, and the pretrained model was used in the experiments. The ProtBERT model was originally introduced in ProtTrans paper [6] and was trained on UniRef and BFD data, containing around 217 million protein sequences (Referred to as Uniref100 and can be downloaded from the official UniProt website [7]. Since this data requires immense computational power, a supercomputer with a TPU was used. The training was done self-supervised, meaning no human annotators worked on the model. Similar to the traditional BERT model used in NLP, the masked language modeling (MLM) objective was used in training with a critical difference in how the sequences are viewed. In the ProtBERT model, sequences are viewed as separate documents; therefore, the objective next sentence prediction was not used, unlike the original NLP model. This model can be fine-tuned for different tasks or used directly for feature extraction from protein sequences.

The model was tested on the DeNovo dataset and compared with other representation methods. In order to use it on our data, some preprocessing has been done. As the first step, similar to DeepViral, sequences have been truncated. In DeepViral, the authors have followed two approaches. They have truncated the sequences as their first 1000 amino acids for their algorithm, and for the replication of the RCNN model, they have truncated the sequences as their first 2000 amino acids. In this work, for the ProtBERT model, sequences were truncated to 256 (total sequence length being 512) since the model is massive, and for larger sequences, it cannot be worked due

to limitations on the hardware. Also, sequences smaller than the cut-off length have been extended to match by concatenating the same sequence until desired length has been reached (Sequence ABCDE becomes ABCDEABC if the desired length is 8). After that, following the documentation from HuggingFace, a sequence is tokenized by amino acids (length as 1). The model works with uppercase sequences; therefore, any lowercase tokens need to be changed to uppercase; also, rare amino acids "U,Z,O,B" need to be mapped to "X". These cases do not appear on any of the datasets that have been used in this work; therefore this step was not necessary for our data. In the end, since the aim of our classifiers is to predict the interaction status between two sequences, a human protein and a viral protein, their representations are concatenated.

A sequential classifier with different setups was used to test the ProtBERT model and compare it with SeqVec model. Hyperparameter tuning on the classifier has been done using the validation datasets, and results are reported in the Experiments section.

4.2.1.2. SeqVec. Similar to ProtBERT for SeqVec, the pre-trained model was used in the experiments with three different datasets for comparison, and this model was not trained from scratch. To create a new model for protein sequences, the authors of the paper that introduces SeqVec [5] utilized the ELMo model that is commonly used in the NLP domain by training it using protein sequence data. A model with two LSTM layers was trained on UniRef50 (similar to the source for ProtBERT, these data can be downloaded from the official UniProt website) for three weeks using 5 GPUs. The original paper compared model performance to other methods on downstream tasks.

Similar to ProtBERT, some preprocessing of the sequences had to be done to test the pretrained SeqVec model on DeNovo, Ebola, and H1N1 datasets. Sequences were at first truncated by taking the first 1000 amino acids following DeepViral's approach. An extension of the sequences is not performed for this model, but additional tests with truncation length have been done. Results of the experiments on cutoff lengths can be found in the Experiments section. After the preprocessing step, sequences are directly given to the model as lists and embeddings are calculated. Following the

calculation, the sequences are concatenated and fed to the classifier model for a pair of human and viral proteins. A sequential classifier is used with the SeqVec model to compare its performance with other methods. Details regarding hyperparameter choice and experiments on different setups of the classifier can be found in the Experiments section.

4.2.2. Graph Based Models

4.2.2.1. Metapath2Vec. Both representation models mentioned before taking the whole sequence into account rather than dividing it into a word form. Similar to sentences in natural language, protein sequences can be possibly divided into sub-pieces that can have a meaning on their own. This work can be done by dividing the sequence directly to n-mers or training a subword-based tokenizer to divide the sequence more meaningfully. Words divided by the BPE model can be utilized in different setups for the representation learning task. Since graphs present an excellent way to show the connections between data points and their relationships, they can be used to model the connections between sequences and the words they contain. With methods like Metapath2Vec [2], it is possible to use these graphs for representation learning. In the next part of this work, this approach has been followed.

Graph-based models can be used for representation learning tasks, such as Attri2Vec, Node2Vec, and Metapath2Vec. Some of these models can be used with heterogeneous graphs, and some cannot. In this work, heterogenous undirected graphs are constructed to use this setup for the pathogen-host interaction prediction task, consisting of different types of nodes and edges (details given below). Since Metapath2Vec can be used for heterogeneous graphs, as in our situation, it was chosen as a suitable method and trained as a graph-based data representation model. The model was trained and tested using all three datasets, DeNovo, H1N1, and Ebola. Types of nodes in the graph are explained in the following section:

- Human (host) protein sequence: Nodes contain sequence ids as an identifier, no additional node features (any data regarding sequence was not added as a feature). Unique nodes for all sequences on the train portion of the used dataset are added to the graph.
- Virus (pathogen) protein sequence: Nodes contain sequence ids as an identifier, no additional node features (any data regarding sequence was not added as a feature). Unique nodes for all sequences on the train portion of the used dataset are added to the graph. Virus and human protein sequence nodes were created as separate nodes since giving them a different type creates the distinction between them.
- Protein word: Nodes containing word ids and no additional node features (any other information regarding words was not added as a feature). Unique nodes for the vocabulary of train data are added to the graph. Vocabulary is created with a BPE tokenizer (ByteLevelBPETokenizer from tokenizers library) trained on SwissProt data and contains 4755 unique sequence pieces or words. All of the vocabulary is added to the graph since the sequences in train sets contain all vocabulary for all sets.

there are different types of edges between these nodes. These edges include:

- Interacts: Edges between human and virus nodes. This edge is added between nodes if their interaction label is 1 on train data.
- Not Interacts: Edges between human and virus nodes. This edge is added between nodes if their interaction label is 0 on train data. This edge was chosen to be added to the graph seperately since no interaction case in our data is not the same as unknown interaction and should be predicted as a negative datapoint. It was also experimentally tested to show that this strategy works better.
- Contains: Edges between human nodes and word nodes, also virus nodes and word nodes. If the sequence has that word in it, the edge is added. Since adding all edges causes sparsity on the graph, these edges were filtered based on TF-IDF scores.

- Similar: Edges between human-human and virus-virus nodes. Similarity scores of sequences are calculated with Needleman-Wunsch algorithm, based on the results, similarity edges were added. Edges were added based on a cut-off value.
- PMI: Edges between word nodes. Since PMI score shows that some words go together (they likely have a meaning together since they appear together often), adding an edge between them can help our graph to become more meaningful. These edges were added based on a cut-off value of PMI scores.

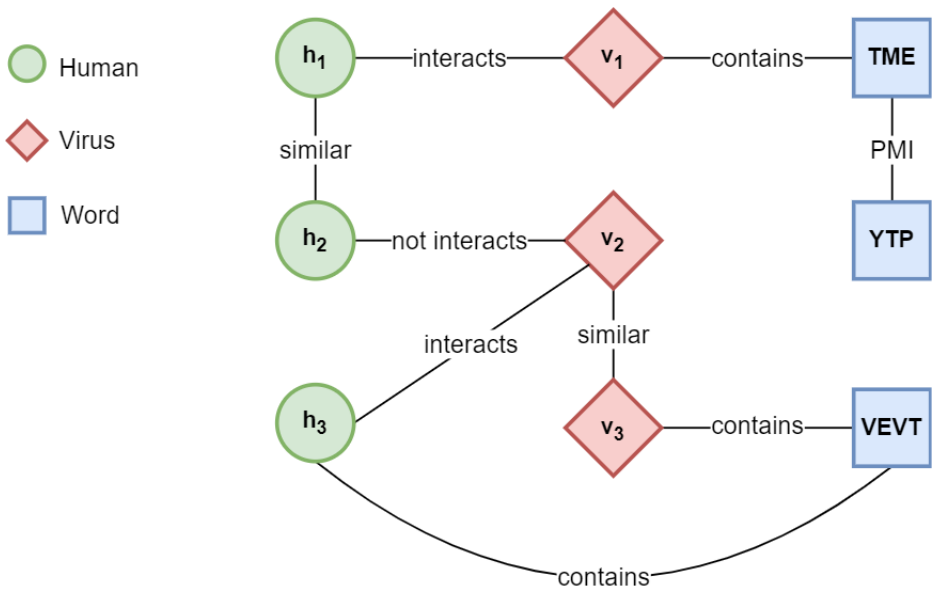

Figure 4.1. Example pathogen-host interaction graph with all edges.

4.2.2.2. Filtering the Edges. Adding all edges, especially between sequences and words, causes the graph to be very sparse. Since the Metapath2vec algorithm works based on random walks, this causes the method to follow many random paths that give little information. Also, it is impossible to distinguish how much weight an edge has on this metapath-based approach since these are not included in walks; we need to know how important that data is for our representation. An analogy with natural language can be made if we give an example from the sequence-word case. The word 'the' appears in many sentences but does not give any information about the meaning. Therefore, if we create a graph with sentences and words, adding edges between almost every sentence and 'the' would only make the graph sparse and complicated. This would make

the graph harder for our algorithm to learn meaningful representations from since it traverses the graph randomly and will stumble upon many edges with 'the' and other sequences-words, missing the important ones.

To avoid the problem with graph traversal, two approaches could be followed. First, we could add weights from our calculated TF-IDF, PMI, and Needleman Wunsch values without filtering any of the edges. This method solves the problem of importance but cannot solve the random walk issue. Also, it creates memory problems since it makes the graph big and sparse, and, challenging to work on. As a second option, we filter out the data by deciding some thresholds for all these edge types and not give weights. The experiments followed this approach to make the computations easier, make the data more readable and meaningful, and avoid memory problems.

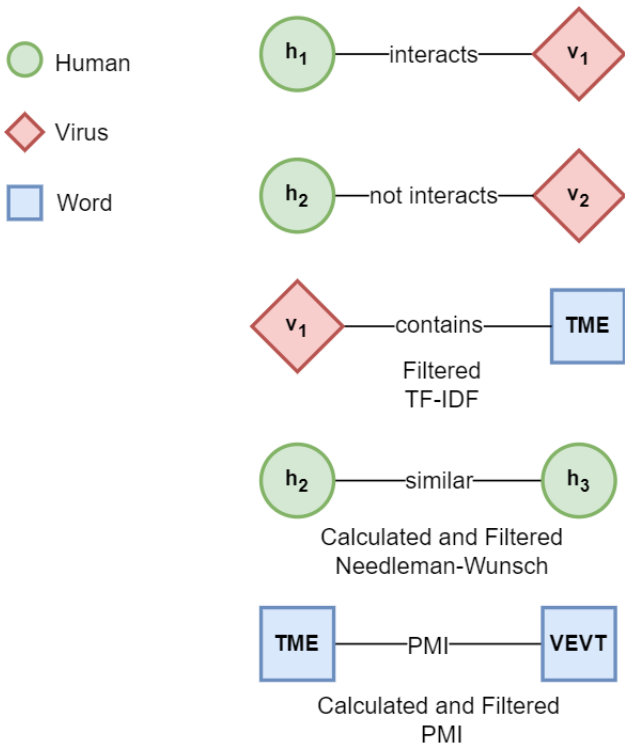

Figure 4.2. All edges and corresponding filtration approaches.

Experiments were done regarding the effects of edge filtering and the importance of adding certain edges to the graph. Numerically it is shown that edge filtering makes

the graph much smaller. Hyperparameter analysis for the thresholds was done using the validation datasets on different setups; details of these experiments can be found in the Experiments section.

4.2.2.3. Creating the Embeddings. Embeddings for train sequences were calculated using the metapath2vec algorithm. The algorithm uses random walks that are meta-path based, meaning that paths to follow are provided before in terms of nodes or edge types depending on the implementation. Using the walks and skip-gram model, representations for the nodes can be learned by the model.

Metapaths showing the way to the random walk algorithm to follow in the graph with the whole setup (containing all node and edge types) are given in figure 4.3

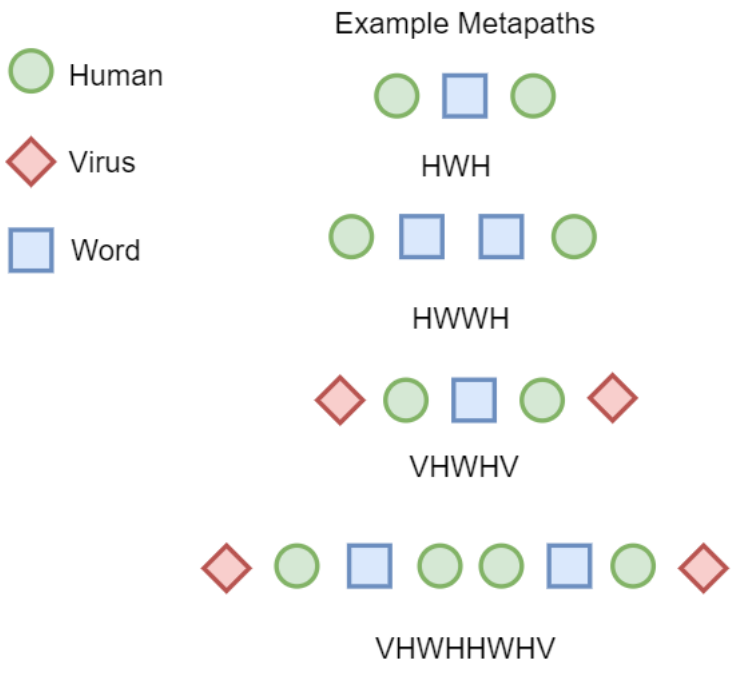

Figure 4.3. Example of different metapaths.

The paths show predefined ways for the algorithm to follow based on nodes. For example, following the "human-human" road will make the algorithm go over a similarity edge since this type of edge connects these two nodes. Another example from

a more complex road such as “human-word-human-virus-human” asks the algorithm to follow this road: Follow the edge between human and word, that is, the edge that shows the human sequence contains that particular word, then follow word to a different human sequence using same kind of edge, then follow human to virus using interacts or not interacts edges and lastly end with a human node. These paths help the algorithm to make connections between different nodes. How we define metapaths changes from implementation to implementation; the one used in this work follows using only the node types for the definition. The only rule is to end the defined metapath with the same node type that the path starts with. The metapaths can be as long as we want them to be.

There is no built-in way to divide the graph and create train and test embeddings separately, as this method initially focuses on clustering. However, there can be different approaches to follow to create embeddings for sequences in the test/validation data for our problem. One approach could be adding all nodes to the model, not providing any information for the test nodes (not adding edges since our nodes do not contain any additional features). However, this could cause some issues with random walks on metapath by creating dead ends that do not give any information. A cleaner approach would be not adding any test nodes to the graph in the first place. However, in this scenario, since nodes belonging to sequences in the test portion of the datasets are not added to the graph, their embeddings can not be calculated directly using the random walks from metapath and word2vec. To tackle this issue, the Needleman-Wunsch algorithm has been used to calculate the similarities between the sequences in the train and test portion of the data and use the generated embeddings for the training dataset to calculate new embeddings for the test set and validation set. It has been done by:

- Embeddings for the train sequences have been generated using Metapath2vec.
- Similarity between train/test sequences has been calculated separately for viruses and humans.
- For a sequence in the test set, most similar sequences in the train set that belong to the same category (pathogen or host) are found.

- The mean value of these sequences that are bigger than a similarity threshold are found and used as the embedding of the test sequence.

and experiments on different cutoff percentile for similarity have also been reported in the next section, as this caused a significant effect on the results.

4.2.2.4. Implementation Details. Data processing was done using Numpy/Pandas. To create graphs and perform the Metapath2vec algorithm, StellarGraph Machine Learning Library was used. The advantage of this library is that it is built upon Tensorflow 2 and can be used if Tensorflow is the choice for development. As a classifier, a sequential Tensorflow model was used. Details of the classification model, training details, and hyperparameter analysis are given in the Experiments section.

4.3. Evaluation Strategy

For the evaluation, a comparison with baseline works and between the experiments of this project was made by calculating Sensitivity, Specificity, Accuracy, PPV, NPV, MCC, AUC and F1 scores. Explanations regarding the used metrics and the calculation details are given in the next section.

4.3.1. Benchmarking Metrics

Many metrics that are used for the analysis are calculated by the confusion matrix, seen on 4.8.

Table 4.8. Confusion matrix.

	Positive Sample	Negative Sample
Positive Prediction	True Positive (TP)	False Positive (FP)
Negative Prediction	False Negative (FN)	True Negative (TN)

Many of the metrics can be calculated via confusion matrix. Among these, sensitivity is defined as the true positive rate calculated by

$$\frac{TP}{TP + FN} \tag{4.1}$$

and shows how good the model is at identifying positive subjects. Together with specificity, it gives more information regarding model performance than accuracy when the data is imbalanced. Specificity is defined as true negative rate. It can be used together with sensitivity and other metrics to give us an idea about the general performance, but it is especially important when we want a model to identify negative subjects, in a situation where the dataset is imlabanced and it is calculated by

$$\frac{TN}{TN + FP} \tag{4.2}$$

using the true negative rate specifically. Similar to sensitivity and specificity, we can calculate negative and positive predictive values to assess the model's performance specifically on negative and positive subjects. These metrics shows the probability that the real result of the test is positive given a positive prediction in PPV

$$\frac{TP}{TP + FP} \tag{4.3}$$

and that the real result of the test is negative, given a negative prediction for NPV

$$\frac{TN}{TN + FN} \tag{4.4}$$

Like sensitivity and specificity, these metrics can be more helpful in a situation where identifying negative or positive subjects is more important. Unlike previous metrics, accuracy, MCC, AUC and F1 shows the general performance of the model. Among these, accuracy is calculated by

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{4.5}$$

with the ratio of correct predictions over all predictions. Even though accuracy is one of the most well-known and widely used metrics in statistics, it can give misleading results when the dataset is not balanced. Metrics like MCC (Matthews correlation coefficient) can be used as a complementary to give us a better idea about the performance. It can be calculated using the calculations on confusion matrix,

$$\frac{TP \times TN – FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4.6}$$

by taking all calculations into account. AUC is another metric that is used frequently in unbalanced datasets. It gives the area under the ROC curve, as given in figure 4.4. It shows how much the model is capable of differentiating different classes. A good model has AUC near 1.

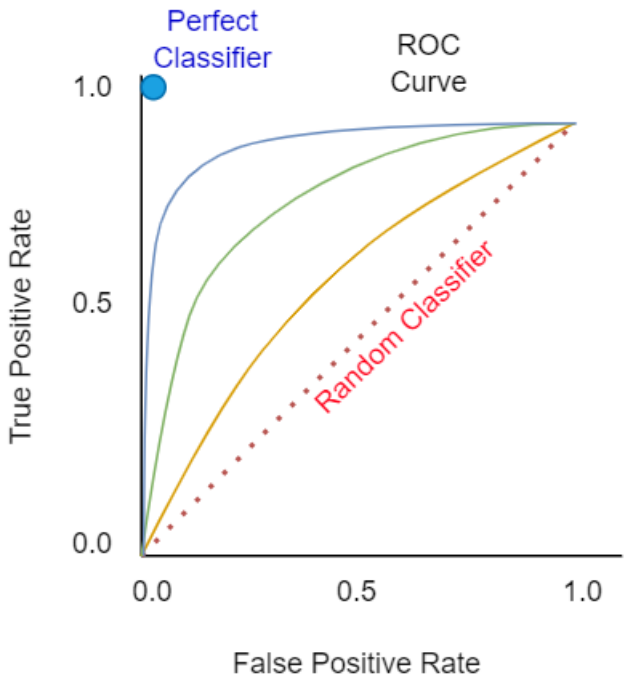

Figure 4.4. Visualization of ROC curve.

Lastly, F1 score is calculated by using both precision and recall, therefore combining them with the formula,

$$\frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.7}$$

as is uses both metrics in the calculation, it can give a general idea about the performance of the model when used together with accuracy.

5. EXPERIMENTS

All experiments reported in the following sections, including data analysis, pre-training, training, validating, and testing the created models, were done using the given setup.

Table 5.1. Environment specifications.

CPU	AMD Ryzen 7 5800HS
GPU	Nvidia GeForce GTX 1650
SSD	Intel SSDPEKNU512GZX
OS	Ubuntu 18.04.4 (Bionic Beaver) LTS
IDE	PyCharm 2021 3.0
Python	3.7
Tensorflow	2.10.0

5.1. Data Representation Models

Representation methods for protein sequences SeqVec and ProtBERT have been compared with each other in various setups to examine their performance on the pathogen-host interaction problem.

5.1.1. Hyperparameter Tuning

Analysis has been done using the validation sets for each of the three datasets for different hyperparameters. Since used datasets contain predefined train and test portions, validation sets are sampled from training data. For each dataset, %10 of the train set is sampled randomly as the validation set and used in the grid search. The primary hyperparameters tuned for both SeqVec and ProtBERT models are sequence length cutoff, number of training epochs for the classifier model, batch size and learning rate, and number of classifier layers. For the ProtBERT model, sequence cutoff values

larger than 256 could not be tested due to memory restrictions caused by the used hardware. For the SeqVec model, cutoff values could be tested up to 1000 due to the model being smaller than ProtBERT. 1000 was chosen as the final cutoff value for the SeqVec model, following the DeepViral approach to compare the results with them. The same classifier model has been used in both setups with the same default hyperparameter options. Default values of the hyperparameters used in the experiments are given for the DeNovo dataset for both models.

Table 5.2. SeqVec model default hyperparameters.

Model Parameter	**Value**
Sequence Cutoff Length	1000
Batch Size	128
Epochs	150
Learning Rate	0.001
Layer Count	4

Table 5.3. ProtBERT model default hyperparameters.

Model Parameter	**Value**
Sequence Cutoff Length	256
Batch Size	128
Epochs	150
Learning Rate	0.001
Layer Count	4

Results have been compared in accuracy, F1, MCC, and AUC scores. Experiments were done with each setup five times, and results and standard deviations were reported for both models.

Table 5.4. SeqVec model hyperparameter analysis results.

		ACC	**F1**	**AUC**	**MCC**
Epochs	**50**	0.900+-0.011	0.899+-0.010	0.963+-0.002	0.802+-0.022
	100	0.899+-0.006	0.898+-0.007	0.959+-0.003	0.799+-0.012
	150	0.907+-0.009	0.912+-0.009	0.962+-0.002	0.815+-0.019
Batch Size	**64**	0.905+-0.004	0.904+-0.004	0.961+-0.003	0.810+-0.008
	128	0.907+-0.009	0.912+-0.009	0.962+-0.002	0.815+-0.019
	256	0.906+-0.008	0.905+-0.008	0.956+-0.003	0.812+-0.017
Layer Count	**2**	0.883+-0.002	0.889+-0.002	0.949+-0.002	0.765+-0.004
	3	0.892+-0.005	0.893+-0.005	0.956+-0.001	0.785+-0.010
	4	0.907+-0.009	0.912+-0.009	0.962+-0.002	0.815+-0.019
Learning Rate	**0.01**	0.898+-0.016	0.903+-0.014	0.965+-0.008	0.797+-0.033
	0.005	0.886+-0.003	0.892+-0.002	0.950+-0.002	0.773+-0.008
	0.001	0.907+-0.009	0.912+-0.009	0.962+-0.002	0.815+-0.019
Seq. Cutoff Length	**64**	0.903+-0.007	0.906+-0.007	0.962+-0.001	0.808+-0.013
	128	0.901+-0.010	0.904+-0.009	0.961+-0.001	0.803+-0.020
	256	0.900+-0.004	0.900+-0.003	0.961+-0.003	0.800+-0.008
	1000	0.907+-0.009	0.912+-0.009	0.962+-0.002	0.815+-0.019

Table 5.5. ProtBERT model hyperparameter analysis results.

		Acc	**F1**	**AUC**	**MCC**
Epochs	**50**	0.817+-0.007	0.809+-0.014	0.909+-0.003	0.639+-0.014
	100	0.837+-0.011	0.832+-0.014	0.921+-0.009	0.677+-0.021
	150	0.856+-0.003	0.849+-0.005	0.937+-0.003	0.713+-0.003
	200	0.866+-0.007	0.873+-0.006	0.938+-0.002	0.732+-0.014
Batch Size	**64**	0.871+-0.013	0.868+-0.014	0.945+-0.006	0.746+-0.024
	128	0.856+-0.003	0.849+-0.005	0.937+-0.003	0.713+-0.003
	256	0.850+-0.011	0.850+-0.012	0.926+-0.004	0.703+-0.023
Layer Count	**2**	0.826+-0.007	0.827+-0.003	0.917+-0.003	0.654+-0.015
	3	0.862+-0.011	0.866+-0.008	0.943+-0.004	0.725+-0.024
	4	0.856+-0.003	0.849+-0.005	0.937+-0.003	0.713+-0.003
Learning Rate	**0.01**	0.735+-0.122	0.758+-0.052	0.806+-0.171	0.481+-0.269
	0.005	0.824+-0.016	0.801+-0.033	0.913+-0.012	0.667+-0.021
	0.001	0.856+-0.003	0.849+-0.005	0.937+-0.003	0.713+-0.003
Seq. Cutoff Value	**64**	0.762+-0.012	0.755+-0.015	0.849+-0.008	0.531+-0.025
	128	0.786+-0.009	0.789+-0.011	0.881+-0.005	0.575+-0.018
	256	0.856+-0.003	0.849+-0.005	0.937+-0.003	0.713+-0.003

From the results, we can see that the most effective hyperparameters are layer count and learning rate for both of the models.

An important observation is that cutoff length does not impact the results as expected for the SeqVec case, even though it is thought to be more critical compared to other hyperparameters. This result can be due to the distribution of sequence lengths in the datasets. Following the analysis in the datasets section, it was seen that most of the sequences in the DeNovo dataset fall between 0-500 range in terms of their lengths. Even among these, almost half of them fall between 0-250. Because of this result, it is not surprising that the cutoff length does not significantly impact the

results in the range of 250-1000. From the experiments with smaller cutoff values, it can be argued that even smaller portions of a sequence can give enough information about its functions; however, more data is needed to support this argument.

5.1.2. Comparison Results

Using the models with the hyperparameters that give the best results on each dataset, the performance of both models is compared. Results are given using the metrics sensitivity, specificity, accuracy, positive predictive value, negative predictive value, MCC, AUC, and F1. For the DeNovo dataset, hyperparameters were chosen according to the tables given in the previous section. A similar analysis was done for Ebola and H1N1 datasets, and the best options were found. Experiments were done five times, and results were reported together with standard deviations.

Table 5.6. ProtBERT and SeqVec results comparison.

Dataset	Model	SN	SP	ACC	PPV	NPV	MCC	AUC	F1
DeNovo	**SeqVec**	0.893 +-0.007	0.939 +-0.007	0.916 +-0.007	0.936 +-0.003	0.898 +-0.007	0.834 +-0.007	0.965 +-0.002	0.914 +-0.004
	ProtBERT	0.780 +-0.034	0.925 +-0.023	0.853 +-0.023	0.914 +-0.007	0.808 +-0.021	0.714 +-0.011	0.940 +-0.001	0.841 +-0.012
Ebola	**SeqVec**	0.950 +-0.011	0.730 +-0.023	0.840 +-0.023	0.779 +-0.013	0.936 +-0.015	0.698 +-0.025	0.942 +-0.004	0.856 +-0.011
	ProtBERT	0.925 +-0.017	0.732 +-0.043	0.828 +-0.043	0.776 +-0.019	0.908 +-0.027	0.670 +-0.034	0.923 +-0.005	0.844 +-0.015
H1N1	**SeqVec**	0.836 +-0.012	0.805 +-0.028	0.820 +-0.028	0.811 +-0.014	0.831 +-0.022	0.642 +-0.028	0.904 +-0.004	0.823 +-0.012
	ProtBERT	0.867 +-0.024	0.687 +-0.040	0.777 +-0.040	0.736 +-0.012	0.839 +-0.020	0.564 +-0.020	0.866 +-0.006	0.795 +-0.008

From the experiments, the SeqVec model gives better results as a representation learning tool for protein sequences in DeNovo and H1N1 datasets. In the Ebola dataset,

results are similar, SeqVec being slightly better. These findings could be attributed to the fact that ProtBERT is a bigger model; therefore, sequences had to be truncated more to use it. However, as could be seen from the hyperparameter results, the SeqVec model still gives better results in the same scenario. This is a surprising outcome since the pre-trained ProtBERT model was trained on more data compared to SeqVec, as explained in the models section. However, it should be noted that these results are from pure pre-trained models used as tools for sequence representation. It is possible to do fine-tuning on both models, which could drastically change the results. Unfortunately, fine-tuning of models could not be done using the default setup.

5.2. Graph Based Models

5.2.1. Hyperparameter Tuning

As explained in the models section, the graph model has many hyperparameters, most of which are related to the creation of the graph by contributing to edge filtering. Unlike data representation models, hyperparameter tuning has been done using validation datasets that were created by randomly sampling ten viruses for the graph model. After sampling ten unique virus sequences, their edges are removed from the graph, including interaction edges and containing edges that indirectly have the data of sequences by showing which words these sequences connect. Instead of sampling the interactions, viruses have been sampled because of the nature of the algorithm that we use in this setup. Since our calculation of validation/test sequence representations depends on finding similar sequences in the train set, it is crucial to use novel viruses in the validation data; otherwise, results would not be as meaningful. This issue has been covered for the test sets when their respective authors created them and to simulate the cases of novel viruses.

Important hyperparameters about edge filtering that were tuned for three datasets and their final values and other parameters are given in the table below.

Table 5.7. Hyperparameters of the graph model and default values.

Category	Hyperparameter	Value
Edge Filtering	PMI	>13
	TF-IDF	max 250
	Needleman-Wunsch	>0.5
Graph Construction	PMI Window Size	20
	Metapath Walk Length	max 250
	Metapath Walks Per Root Node	3
	Word2Vec Vector Size	256
	Word2Vec Window Size	10
BPE	Vocab Size	5000
	Min Frequency	2
Classifier	Epochs	300
	Batch Size	128
	Learning Rate	0.01
	Layer Count	4
Testing	Similarity Cutoff	>0.99

Among the reported parameters, the most impactful ones are in the edge filtering category. The calculations based on PMI, TF-IDF, and Needleman-Wunsch algorithm decide whether to add an edge to the graph or not. Based on the PMI cutoff, edges are added between words, symbolizing that they frequently appear together. The max PMI value is 20.087, and when we look at the trend, we see that values drop drastically. For example, after the first 20 values, the PMI value drops to 16. While eye-checking

these results, it was also seen that many values drop between 13 and less; therefore, 13 was chosen as a default value, and other values were also tested as the cutoff. In this case, we choose 13 as the cutoff, reducing the corresponding edge count significantly from 22610025 to 5215. TF-IDF values are calculated for each word treating sequences as documents and words as terms. After the calculation, scores are sorted, and the highest scored 250 words are chosen. Edges between sequences and these chosen words are added to the graph. This makes the corresponding edge count almost 1/3 of the original. The effects of edge filtering are shown in a more detailed way in the 5.2.2. Lastly, the Needleman-Wunsch algorithm calculates the similarity edges between two sequences. Based on these scores, if the similarity between two sequences is greater than 50 (max similarity is 100), an edge is added between them. These parameters have been analyzed on the validation set with novel viruses created from the train set. For all parameters, default values were used except for the 'Metapath walks per root node.' For that parameter, one is used to make experiments faster since incrementing this parameter increase runtime significantly. Embeddings for the validation set are calculated in the same way as the test set by using the created embeddings of train sequences based on similarity, explained in a more detailed way in the models section. The analysis results on the DeNovo dataset are given below; analyses on other datasets are added to the Appendix.

Table 5.8. Hyperparameter analysis on the graph model for DeNovo dataset.

		ACC	**F1**	**AUC**	**MCC**
PMI	**>11**	0.828 +-0.029	0.797 +-0.038	0.948 +-0.008	0.687 +-0.052
	>12	0.824 +-0.029	0.794 +-0.039	0.952 +-0.008	0.677 +-0.054
	>13	0.828 +-0.014	0.793 +-0.020	0.961 +-0.006	0.695 +-0.025
	>14	0.859 +-0.020	0.838 +-0.029	0.957 +-0.007	0.744 +-0.032
TF-IDF	**max 50**	0.826 +-0.032	0.793 +-0.043	0.942 +-0.014	0.687 +-0.058
	max 250	0.828 +-0.014	0.793 +-0.020	0.961 +-0.006	0.695 +-0.025
	max 500	0.843 +-0.028	0.817 +-0.040	0.956 +-0.008	0.717 +-0.047
Needleman Wunsch	**>35**	0.835 +-0.034	0.801 +-0.049	0.899 +-0.023	0.708 +-0.057
	>50	0.828 +-0.014	0.793 +-0.020	0.961 +-0.006	0.695 +-0.025
	>75	0.826 +-0.009	0.792 +-0.015	0.939 +-0.007	0.690 +-0.013
	>90	0.821 +-0.023	0.789 +-0.035	0.934 +-0.018	0.673 +-0.040

Using edge filtering improved the results and sped up the runtime. For the DeNovo dataset, statistics of the created graph with edge filtering using the best setups and the graph with all edges are given in 5.9. The number of total edges reduces significantly with the filtered graph.

Table 5.9. Statistics of graph without edge filtering.

Nodes	**Count**
Human	2263
Virus	385
Word	4755
Edges	
Sequence - Contains - Word	497478
Human - Interacts - Virus	4545
Human - Not Interacts - Virus	4233
Human - Similar - Human	5121169
Virus - Similar - Virus	148225
Word - PMI - Word	22610025

Table 5.10. Statistics of graph with edge filtering.

Nodes	**Count**
Human	2263
Virus	385
Word	4755
Edges	
Human - Contains - Word	131851
Virus - Contains - Word	18913
Human - Interacts - Virus	4545
Human - Not Interacts - Virus	4233
Human - Similar - Human	951
Virus - Similar - Virus	311
Word - PMI - Word	5215

Since the construction of the graph can tell us which structures are more important (in terms of different edges), different versions of the graphs have also been tested on all datasets, and results have been reported to assess the importance of certain features in 5.2.2.

5.2.2. Analysis on Graph Structure

Following the hyperparameter tuning, another critical issue regarding the graph is whether to add certain types of edges or nodes to the graph. Various experiments were done to see how adding specific nodes/edges affected the result. In order to make comparisons, different graphs were created with different nodes and edges. Created graphs are:

- Setup 1: Graph that only contains interacts and not interacts edges and sequence nodes, as given in figure 5.1. This graph does not contain any additional information regarding similarities between sequences, and it also does not contain any word nodes or any edges between them. It only shows which sequences interact and which ones do not.

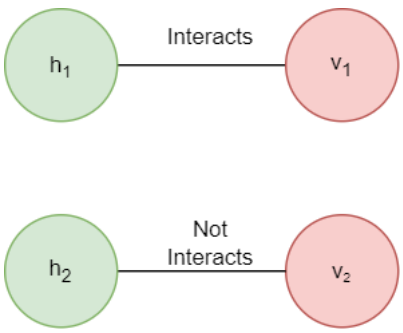

Figure 5.1. Graph setup with no additional features.

- Setup 2: The graph in this setup has word nodes as well as protein nodes, as shown in figure 5.2. It contains edges regarding words, including contain edges that show which sequences contain which words, PMI edges that show connection between words. This setup does not have similarity edges.

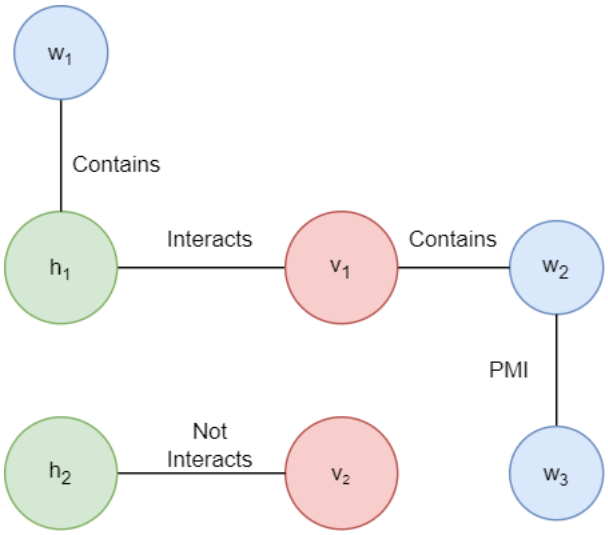

Figure 5.2. Graph setup with protein word nodes and associated edges.

- Setup 3: The graph in this setup does have sequence nodes; however, it does not have word nodes or edges regarding these nodes (PMI and includes edges), as seen in 5.3. It has similarity edges between human-human and virus-virus sequence nodes. It is created to show much the similarities are useful without words.

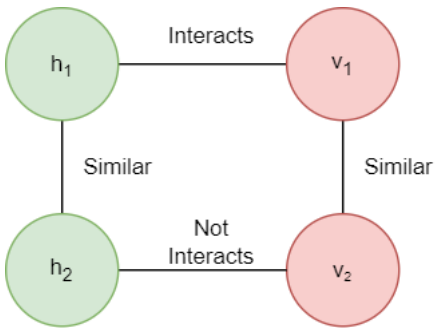

Figure 5.3. Graph setup with similarity edges.

- Setup 4: The graph in this setup has word nodes as well as protein nodes. It also has similarity edges between sequences and contain edges between words and sequences, a simple example given in 5.4. It does not have the PMI edges, which are edges between words. This graph shows how much and if adding these extra edges between words improves the result.

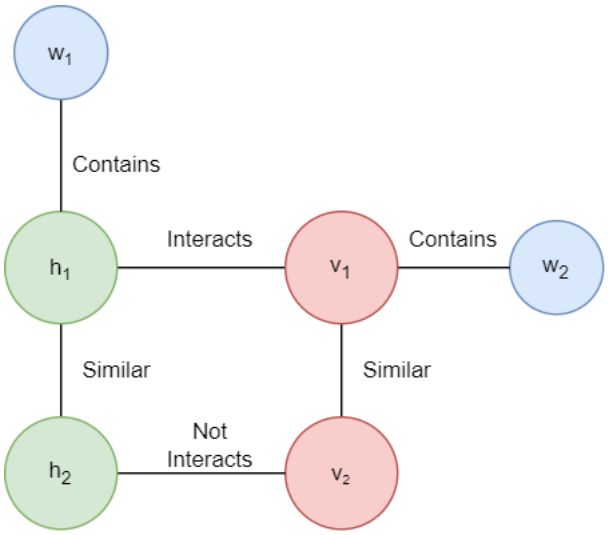

Figure 5.4. Graph setup only with word contain edges and similarities.

- Setup 5: Graph with the usual setup consists of all original nodes and edges, given in figure 5.5. These include protein and word nodes, interacts and not interacts edges, contains, PMI, and similarity edges.

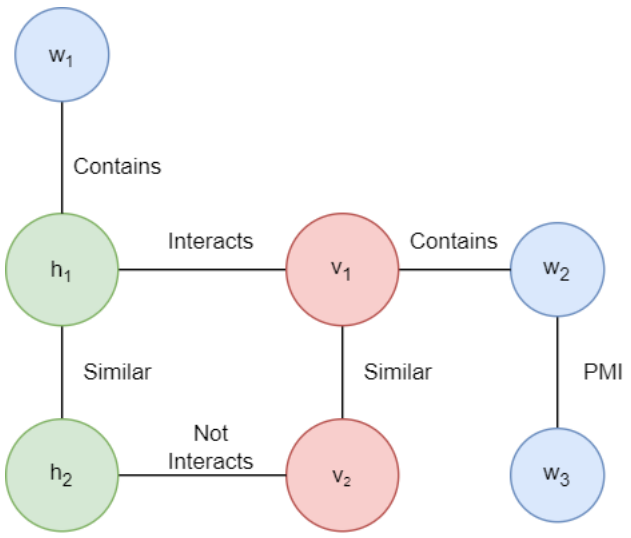

Figure 5.5. Final graph setup that contains all edges.

and these graphs are tested on different datasets.

Table 5.11. Comparison of graph based models on different setups.

	Graph Setup	SN	SP	ACC	PPV	NPV	MCC	AUC	F1
DeNovo	**Base**	0.733	0.917	0.825	0.900	0.775	0.663	0.898	0.807
	+C+P-S	0.878	0.941	0.909	0.937	0.885	0.821	0.957	0.906
	-C-P+S	0.855	0.951	0.903	0.946	0.868	0.811	0.960	0.904
	+C-P+S	0.919	0.945	0.930	0.944	0.923	0.867	0.972	0.932
	All	0.900	0.967	0.933	0.964	0.906	0.869	0.973	0.931
Ebola	**Base**	0.932	0.634	0.783	0.719	0.903	0.594	0.892	0.811
	+C+P-S	0.958	0.628	0.793	0.720	0.938	0.621	0.910	0.822
	-C-P+S	0.948	0.660	0.804	0.736	0.926	0.628	0.910	0.829
	+C-P+S	0.954	0.664	0.809	0.739	0.936	0.646	0.912	0.833
	All	0.952	0.704	0.826	0.762	0.932	0.673	0.915	0.845
H1N1	**Base**	0.832	0.635	0.734	0.696	0.791	0.478	0.805	0.758
	+C+P-S	0.906	0.625	0.765	0.707	0.870	0.554	0.824	0.794
	-C-P+S	0.894	0.640	0.767	0.713	0.859	0.553	0.823	0.793
	+C-P+S	0.912	0.626	0.769	0.709	0.878	0.563	0.821	0.798
	All	0.913	0.643	0.778	0.719	0.882	0.605	0.842	0.805

From the results, it is seen that adding additional edges for both sequence similarity and words improve the results on all three datasets, especially in terms of accuracy, AUC, and F1 scores. Even though it is excepted that the similarity edges improve scores since they carry information regarding sequences, it is notable for showing that adding word nodes and edges improves the results as much as adding similarities. They have even more impact in some cases, showing that the words can carry some meaning and information regarding the sequences to which they belong, which is helpful for the model to learn the representation. Adding PMI edges has a negligible impact on the results, showing that another approach can be considered to add edges between words in future work.

5.2.3. Analysis on Test Embeddings Similarity Cutoff Value

Since test and validation embeddings can not be created during the training to prevent data leakage, they are created after the embeddings for train sequences are created with the Metapath2Vec algorithm using Needleman-Wunsch similarities, as explained in the models section. A very impactful decision on the final results is the similarity cutoff value when test embeddings are created. This value corresponds to the similarity cutoff where train embeddings are used when creating the test embeddings. Experiments were done to show the difference in results based on this value regarding the accuracy, given in 5.6 and F1 scores, given in 5.7.

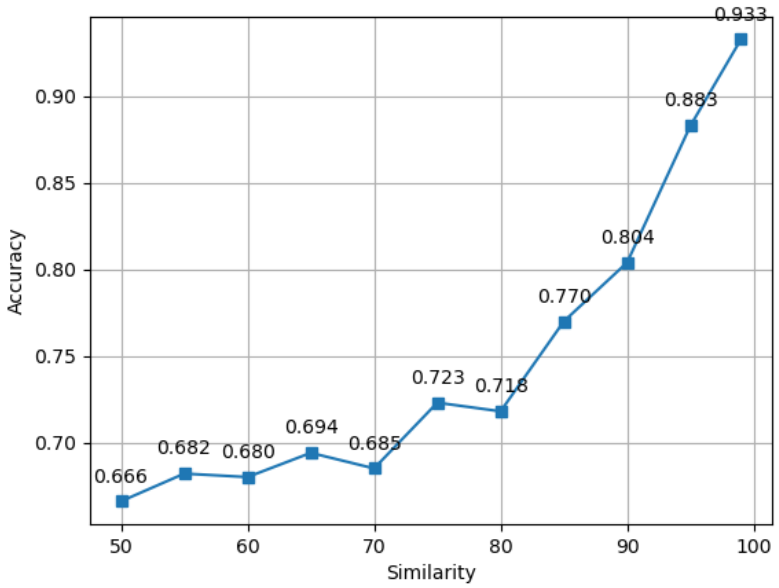

Figure 5.6. Similarity cutoff value vs accuracy graph.

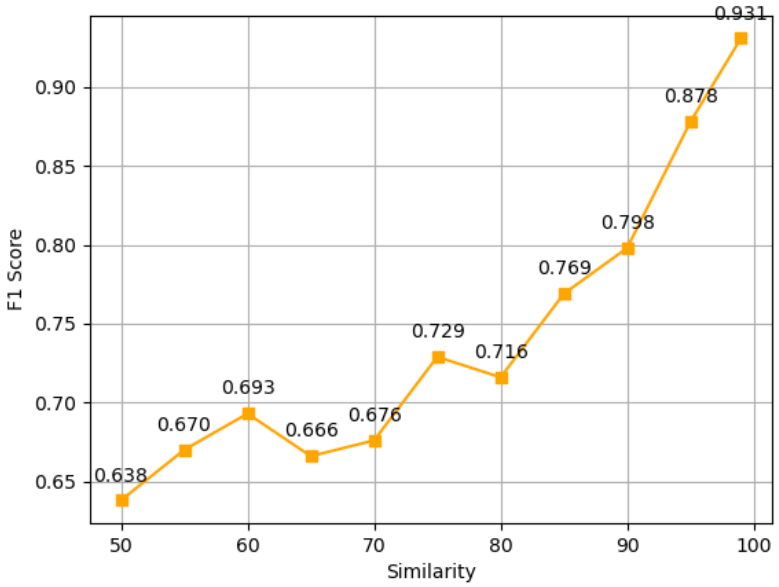

Figure 5.7. Similarity cutoff value vs F1 score graph.

Results from the graphs 5.6, 5.7 show that there is a significant drop in all metrics after 0.9 to 0.95 similarity. This can show that the model can be improved in robustness, or an alternative approach can be tested for embedding calculation. One possibility is overfitting, which could explain the results of the graph model being generally better in DeNovo set than other sets since it is smaller than them. Another possibility is that since we are taking the unweighted mean of different embeddings to calculate another embedding, this could change the meaning and cause created embeddings to lose meaning. Different approaches can be tested for future work, such as getting the embeddings of words and using them to create test embeddings.

5.2.4. Approximate Randomization Tests

Approximate randomization tests were run on graphs to assess statistical value of the results. We can use these tests in deep learning to see the significance of the performance of a model on a particular task or dataset. For this task, tests were run on the three datasets, DeNovo, Ebola and H1N1. Tests are done following these steps:

- A model is trained on the training set.
- Performance of the trained model is assessed in test set.
- Large number of shuffled versions of the test set is generated by randomly assigning the labels. For all sets, 10000 random versions of the test sets were created.
- Predictions on each shuffled test set is done, metrics are calculated.
- The performance of the original model on the true test set is compared to the metrics generated by the shuffled models.
- The p-value is calculated as the proportion of shuffled versions that have an accuracy (or any other metric) that is greater than or equal to the one obtained from the original test set.
- A low p-value that is less than a significance level shows that it is unlikely to observe the performance of the original model, if the null hypothesis (the model has learned nothing from the data) were true.
- Showing that we reject the null hypothesis and accept the alternative hypothesis can show the model has learned something meaningful from the data.

and P values for each dataset are calculated based on accuracy value for the graph model that has all edges and nodes. It was seen that all values are below the significance threshold (decided as 0.01), and in DeNovo and Ebola datasets, the value is calculted 0.00, showing that the results from our models are significant.

5.3. Comparison Results with Previous Methods

The performance of the created models has been tested on three different datasets from two sources, namely DeNovo, Ebola, and H1N1 sets.

5.3.1. DeNovo Dataset

The DeepViral model has been used as the baseline for the experiments on this DeNovo dataset. In the original paper, the authors have introduced two models. One is a sequence-based model, which only uses sequence data of viruses and human proteins.

Another model is a phenotype model, where they use the DL2Vec algorithm [26] to generate feature embeddings using ontologies gathered from PhenomeNET [48]. These models have been used separately and jointly in their papers. They have tested the models in four different setups: sequence model, sequence model with human phenotype embeddings, sequence model with virus phenotype embeddings, the joint model, which has embeddings for both viral and host protein, as well as sequence information. Since models in this work use only sequence information, the sequence model of DeepViral has been taken as a baseline, but results were compared with all models. In the same paper, they tested the data with the RCNN model and reported the results. Also, some previous works used this dataset, and their results are added to the comparison table.

The performance of the tested models with previous methods that use the same dataset is provided in the following table. From the results in table 5.12, it is seen that the joint graph model gives similar results that are slightly better and slightly worse in some metrics compared to the sequence model of DeepViral. It is seen that even though the joint model of DeepViral is superior to other methods, showing that ontology information is meaningful and improves the results on this problem, results are not strikingly different. These results indicate that sequence information can be used to get meaningful models for this task.

Table 5.12. Performance of different methods on DeNovo dataset.

Method	SN	SP	ACC	PPV	NPV	MCC	AUC	F1
DeNovo	0.807	0.830	0.819	NA	NA	NA	NA	NA
VirusHostPPI	0.800	0.889	0.844	0.878	0.816	0.692	0.897	NA
Alguwzizani et al.'s SVM	0.863	0.865	0.864	0.865	0.863	0.729	0.926	NA
Doc2Vec+RF	0.903	0.961	0.932	0.959	0.907	0.866	0.981	0.930
RCNN	0.898	0.955	0.927	0.953	0.904	0.857	0.974	0.974
DeepViral (seq)	0.893	0.968	0.931	0.966	0.901	0.865	0.960	0.928
DeepViral(seq+hum emb.)	0.884	0.962	0.923	0.959	0.892	0.849	0.955	0.920
DeepViral(seq+viral emb.)	0.882	0.972	0.927	0.969	0.892	0.859	0.967	0.924
DeepViral (joint)	0.902	0.975	0.939	0.974	0.909	0.881	0.976	0.936
SeqVec+Sequential	0.893	0.939	0.916	0.936	0.898	0.834	0.965	0.914
ProtBERT+Sequential	0.780	0.925	0.853	0.914	0.808	0.714	0.940	0.841
Metapath2vec(base)	0.733	0.917	0.825	0.900	0.775	0.663	0.898	0.807
Metapath2vec(word)	0.878	0.941	0.909	0.937	0.885	0.821	0.957	0.906
Metapath2vec(similarity)	0.855	0.951	0.903	0.946	0.868	0.811	0.960	0.904
Metapath2vec(word+sim.)	0.900	0.967	0.933	0.964	0.906	0.869	0.973	0.931

5.3.2. Ebola and H1N1 Datasets

Additional experiments were done on the created models using two more datasets, Ebola and H1N1, created by Zhou et al. [14] ProtBERT and SeqVec models, as well as graphs were created in the same way as explained in the previous sections. Created graphs include the same types of nodes and edges and the same filtering methods. Promising results were achieved compared to the original paper; however, the method seems lacking compared to the MotifTransformer method. One of the critical reasons for this result is that MotifTransformers use a transfer learning approach, consequently, more data. As they also showed in their work, this approach improves the results. Details of the created graphs and filtering parameters can be found in Appendix A. Performance comparison tables for both datasets are given below.

Table 5.13. Performance of different methods on Ebola dataset.

Method	SN	SP	ACC	PPV	NPV	MCC	AUC	F1	AUPR
Zhou et. al.'s SVM	0.906	0.653	0.780	0.723	0.875	0.579	0.867	0.760	NA
MotifTransformer	NA	NA	NA	NA	NA	NA	0.912	0.860	0.953
MotifTrans.+UPT	NA	NA	NA	NA	NA	NA	0.953	0.904	0.961
MotifTrans.+UPT+SPT	NA	NA	NA	NA	NA	NA	0.968	0.896	0.974
SeqVec+Sequential	0.945	0.731	0.838	0.779	0.930	0.693	0.838	0.854	NA
ProtBERT+Sequential	0.908	0.762	0.835	0.795	0.893	0.679	0.835	0.847	NA
Metapath2vec(base)	0.932	0.634	0.783	0.719	0.903	0.594	0.892	0.811	NA
Metapath2vec(word)	0.958	0.628	0.793	0.720	0.938	0.621	0.910	0.822	NA
Metapath2vec(similarity)	0.948	0.660	0.804	0.736	0.926	0.628	0.910	0.829	NA
Metapath2vec(word+sim.)	0.952	0.704	0.826	0.762	0.932	0.673	0.915	0.845	NA

Table 5.14. Performance of different methods on H1N1 dataset.

Method	SN	SP	ACC	PPV	NPV	MCC	AUC	F1	AUPR
Zhou et. al.'s SVM	0.889	0.658	0.774	0.722	0.856	0.564	0.884	0.762	NA
MotifTransformer	NA	NA	NA	NA	NA	NA	0.903	0.844	0.898
MotifTrans.+UPT	NA	NA	NA	NA	NA	NA	0.908	0.855	0.903
MotifTrans.+UPT+SPT	NA	NA	NA	NA	NA	NA	0.945	0.865	0.948
SeqVec+Sequential	0.836	0.805	0.820	0.811	0.831	0.642	0.904	0.823	NA
ProtBERT+Sequential	0.867	0.687	0.777	0.736	0.839	0.564	0.866	0.795	NA
Metapath2vec(base)	0.832	0.635	0.734	0.696	0.791	0.478	0.805	0.758	NA
Metapath2vec(word)	0.906	0.625	0.765	0.707	0.870	0.554	0.824	0.794	NA
Metapath2vec(similarity)	0.894	0.640	0.767	0.713	0.859	0.553	0.823	0.793	NA
Metapath2vec(word+sim.)	0.913	0.643	0.778	0.719	0.882	0.605	0.842	0.805	NA

6. DISCUSSION AND FUTURE WORK

The central claim of this thesis is to show if sequences can be treated as sentences, and dividing them into words could give an idea about the meaning or structure of the sequence they belong. With BPE, sequences are divided into words, and these words are integrated into a graph model. To show to impact of the words, different setups were created. These setups contained graphs with only interaction information, graphs with similarity edges between proteins calculated using the Needleman-Wunsch algorithm, and graphs with word nodes with different edges. Word-related edges include edges that show if a sequence contains a word and PMI edges that show if the words go together, meaning they appear in the same sequence frequently and possibly have meaning together. These edges were filtered using TF-IDF and a cutoff value for PMI. Filtering the edges decreased the sparsity of the graph, and with the hyperparameter analysis, this decrease in edge count positively impacted the performance of all datasets. This process also positively impacted runtime, as expected, since using the full graph without any filtering caused memory problems. Effects of word edges were also compared by creating a setup that does not contain word-word edge PMI. This setup showed that adding PMI edges is less effective than adding contain edges (word-sequence edges). This result can be studied further in future works to test alternative approaches to creating word-word edges. Since adding similarity edges between proteins is generally a more established concept, it was expected that adding these edges to affect the results positively; however, we showed that adding word edges caused the same effect. Adding only similarity or only word-related edges made a similar increase in results; adding both together made a more significant effect. This result showed that both types of information regarding sequences could be used together to get more idea about a protein from its sequence. As a future work, protein words can be examined more profoundly regarding their possible meaning. For example, embeddings created for sequence nodes from the graph setting can be compared to word embeddings to see if we can find any connection, and experiments regarding explainability can be performed. Another area to use the word embeddings can be the calculation of test

embeddings. In the current setup, test embeddings are calculated using similar train embeddings by finding the most similar ones and getting their mean. This method can be replaced by using the sequence's word embeddings to create new test embeddings. If this method can achieve similar success, we can further demonstrate that words indeed carry more meaning regarding the sequence they belong to.

In the first set of experiments, representation learning models ProtBERT and SeqVec were compared on three different datasets using different setups. Compared with the graph model, they give similar or worse results. Since these models were trained using UniRef50 and UniRef100, massive datasets compared to the ones we use in the graph setup, we further showed the potential of the graph model. When these models were compared, results showed that SeqVec performed better than ProtBERT. This finding was surprising for a couple of reasons. First, the BERT model is a prevalent representation model in the NLP domain since using attention with transformers improves the results. The equivalent BERT model on the NLP domain gives state-of-the-art results in almost every related problem and is seen as superior to the ELMo model, even though both models are used in all tasks and give excellent results. The second reason is that BERT is a bigger model than ELMo, and their consequent versions for protein sequences ProtBERT and SeqVec follow the same trend. Being a bigger model, it is expected ProtBERT to perform better. The last reason is the data used to train these models. While SeqVec was trained using UniRef50, a massive database, ProtBERT uses UniRef100 for training, which is much bigger. It is expected that this difference will make a change in results. A reason for the results favoring SeqVec could be the cutoff length for given sequences. This factor was eliminated to make a fair comparison by creating models with the same protein lengths. Results came up similarly, still, SeqVec performed slightly better. Another reason could be the setup. It is seen that ProtBERT gives better performance on the H1N1 dataset compared to others, showing that it could be purely related to data, and more experiments on different datasets are needed to make a definitive conclusion. As it will be discussed later, the datasets we used in the work are small, and they are balanced, unlike the real data. Another reason could be the usage of embeddings. In this work, after calculating the embeddings for

both sequences, they are concatenated and fed to the classifier model in all setups. A different setup may cause the ProtBERT model to perform better and can be tested in future work.

It is worth noting that the datasets we have used in this work have a downside. Like most of the datasets used for pathogen-host interaction problems, negative portion of the sets were created using random sampling. Unfortunately, this method does not guarantee that the negative points are valid. Another downside regarding the negative set is the size. In the real world, negative interactions are much more common than positive ones. In an actual real-life set, the opposing set should be much bigger than a positive set. In our datasets, the negative dataset is created the same size as the positive set. Because of this reason, the performance of the models on these sets may be an overestimation of the actual performance. In a dataset that simulates a situation closer to a real-life scenario, metrics like PPV (Positive Predictive Value) would be more critical than in our scenario, where we valued metrics such as accuracy and F1 score more because of the partition of our datasets. In future works, different portions for the negative dataset can be created, and datasets can be expanded to create a situation that simulates reality more.

A critical feature of all models used in the thesis is that they utilize only the sequence information with the interaction information. Comparing the results of state-of-the-art models, using extra data, such as ontology information or using transfer learning, could make improvements. Integrating ontology data into the graph model can be considered as a future work.

7. CONCLUSION

In this thesis, we compared different representation methods for protein sequences, used them in pathogen-host interaction problem, and created an alternative approach that can give competitive results. We used various representation methods in the experiments, SeqVec, ProtBERT, and the graph model with Metapath2Vec.

We used representation learning methods with pre-trained models in the first part of the experiments and compared their performance. A surprising result of these experiments was that the SeqVec model was superior to the ProtBERT model with the given setup. In the second part, a new model that uses graph-based learning was introduced. This graph is created using the word data by dividing sequences into words, interaction data, and data regarding similarities of sequences. A significant result of this process is the division of protein sequences. With the graph model, the hypothesis was to treat sequences as sentences and divide them into words using a tokenization algorithm. It was expected to see that parts of the sequence, named as words, give an idea about the whole sequence. To test this hypothesis, we used words in the graph instead of the complete sequence information of proteins.

From the experiments, adding word information to the graph increased the performance, showing that words carry information regarding the protein sequence they belong. Also, the graph method performed better than ProtBERT and SeqVec models in the DeNovo case, even though it uses less data than these methods. The graph method gives comparable results in other sets, Ebola and H1N1. Another comparison was made by adding the sequence similarities to the graph to see if it made more difference in the results. It was shown that similarity information and words are both valuable and positively impact the results. From the results, we can argue that graph-based methods could be an excellent alternative for representation learning.

We tested all methods in three datasets created by previous works, DeNovo, Ebola, and H1N1, to use the same partitions as them and compare the results. The earlier works that use these datasets include experiments using extra data like ontologies or having an additional pre-training objective. The comparisons for all methods in all sets showed that all models could achieve comparable or better results with the previous methods that contain sequence information; however, methods that include extra data are superior.

REFERENCES

1. Chen, H., J. Shen, L. Wang and J. Song, "A Framework Towards Data Analytics on Host-Pathogen Protein-Protein Interactions", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 11, No. 11, pp. 4667–4679, 2020.

2. Dong, Y., N. V. Chawla and A. Swami, "metapath2vec: Scalable Representation Learning for Heterogeneous Networks", *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 135–144, Halifax NS Canada, August 2017.

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

4. Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, "Deep Contextualized Word Representations", arXiv:abs/1802.05365, 2018.

5. Heinzinger, M., A. Elnaggar, Y. Wang, C. Dallago, D. Nechaev, F. Matthes and B. Rost, "Modeling Aspects of the Language of Life through Transfer-Learning Protein Sequences", *BMC Bioinformatics*, Vol. 20, No. 1, pp. 1–17, 2019.

6. 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 Life's Code Through Self-Supervised Deep Learning and High Performance Computing", arXiv:abs/2007.06225, 2020.

7. Apweiler, R., A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J Martin, D. A Natale, C. O'Donovan, N. Redaschi and L.-S. L Yeh, "Uniprot: The Universal Protein Knowledgebase", *Nucleic Acids Research*, Vol. 32, No. 90001, pp. 115–119, 2004.

8. Sun, P. D., C. E. Foster and J. C. Boyington, "Overview of Protein Structural and Functional Folds", *Current Protocols in Protein Science*, Vol. 35, No. 1, pp. 1711–17.1, 2004.

9. Casadevall, A. and L.-a. Pirofski, "Host-Pathogen Interactions: Basic Concepts of Microbial Commensalism, Colonization, Infection, and Disease", *Infection and Immunity*, Vol. 68, No. 12, pp. 6511–6518, 2000.

10. Sayers, E. W., E. E. Bolton, J. R. Brister, K. Canese, J. Chan, D. C. Comeau, R. Connor, K. Funk, C. Kelly, S. Kim, T. Madej, A. Marcher-Bauer, C. Lanczycki, S. Lathrop, Z. Lu, F. Thibaud-Nissen, T. Murphy, L. Phan, Y. Skripchenko, T. Tse, J. Wang, R. Williams, B. W Trawick, K. D Pruitt and S. T Sherry, "Database Resources of the National Center for Biotechnology Information", *Nucleic Acids Research*, Vol. 43, No. D1, pp. D13–D21, Jan 2022.

11. Berman, H., K. Henrick and H. Nakamura, "Announcing the Worldwide Protein Data Bank", *Nature Structural amp; Molecular Biology*, Vol. 10, No. 12, pp. 980–980, 2003.

12. Eid, F.-E., M. ElHefnawi and L. S. Heath, "DeNovo: Virus-Host Sequence-Based Protein-Protein Interaction Prediction", *Bioinformatics*, Vol. 32, No. 8, pp. 1144–1150, 12 2015.

13. Orchard, S., S. Kerrien, S. Abbani, B. Aranda, J. Bhate, S. Bidwell, A. Bridge, L. Briganti, F. S. Brinkman, G. Cesareni, A. Chatr-aryamontri, E. Chautard, C. Chen, M. Dumousseau, J. Goll, R. E. W. Hancock, L. I. Hanrick, I. Jurisica, J. Khadake, D. J. Lynn, U. Mahadevan, L. Perfetto, A. Raghunath, S. Ricard-Blum, B. Roechert, L. Salwinski, V. Stümpflen, M. Tyers, P. Uetz, I. Xenarios and H. Hermjakob, "Protein Interaction Data Curation: The International Molecular Exchange (imex) Consortium", *Nature Methods*, Vol. 9, No. 4, pp. 345–350, 2012.

14. Zhou, X., B. Park, D. Choi and K. Han, "A Generalized Approach to Predicting

Protein-Protein Interactions Between Virus and Host", *BMC Genomics*, Vol. 19, No. S6, pp. 568–568, 2018.

15. del Toro, N., A. Shrivastava, E. Ragueneau, B. Meldal, C. Combe, E. Barrera, L. Perfetto, K. How, P. Ratan, G. Shirodkar, O. Lu, B. Meszaros, X. Watkins, S. Pundir, L. Licata, M. Iannuccelli, M. Pellegrini, M. Jesus Martin, S. Panni, M. Duesbury, S. D Vallet, J. Rappsilber, S. Ricard-Blum, G. Caserini, L. Salwinski, S. Orchard, P. Porras, K. Panneerselvam and H. Hermjakob, "The Intact Database: Efficient Access to Fine-Grained Molecular Interaction Data", *Nucleic Acids Research*, Vol. 50, No. D1, pp. 648–653, 2021.

16. Sennrich, R., B. Haddow and A. Birch, "Neural Machine Translation of Rare Words with Subword Units", arXiv:abs/1508.07909, 2015.

17. Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Guger, M. Drame, Q. Lhoest and A. M. Rush, "Transformers: State-of-the-art natural language processing", *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020.

18. Perozzi, B., R. Al-Rfou and S. Skiena, "DeepWalk: online learning of social representations", *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 701–710, New York USA, August 2014.

19. Grover, A. and J. Leskovec, "node2vec: Scalable Feature Learning for Networks", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 855–864, New York USA, August 2016.

20. Bhatta, J., D. Shrestha, S. Nepal, S. Pandey and S. Koirala, "Efficient Estimation of Nepali Word Representations in Vector Space", *Journal of Innovations in Engineering Education*, Vol. 3, No. 1, pp. 71–77, 2020.

21. Sammut, C. and G. I. Webb (Editors), *TF–IDF*, pp. 986–987, Springer US, Boston, MA, 2010.

22. Nourani, E., F. Khunjush and S. Durmus, "Computational Approaches for Prediction of Pathogen-Host Protein-Protein Interactions", *Frontiers in Microbiology*, Vol. 6, pp. 94–94, 2015.

23. Tastan, O., Y. Qi, J. G. Carbonell and J. Klein-Seetharaman, "Prediction of Interactions Between HIV-1 and Human Proteins by Information Integration", *Biocomputing 2009*, pp. 516–527, 2008.

24. Consortium, G. O., "The Gene Ontology Resource: 20 Years and Still Going Strong", *Nucleic Acids Research*, Vol. 47, No. D1, pp. 330–338, 2018.

25. Liu-Wei, W., S. Kafkas, J. Chen, N. J. Dimonaco, J. Tegnér and R. Hoehndorf, "Deepviral: Prediction of Novel Virus-Host Interactions from Protein Sequences and Infectious Disease Phenotypes", *Bioinformatics*, Vol. 37, No. 17, pp. 2722–2729, 2021.

26. Chen, J., A. Althagafi and R. Hoehndorf, "Predicting Candidate Genes from Phenotypes, Functions and Anatomical Site of Expression", *Bioinformatics*, Vol. 37, No. 6, pp. 853–860, 2020.

27. Kumar, R. and B. Nanduri, "HPIDB - a Unified Resource for Host-Pathogen Interactions", *BMC Bioinformatics*, Vol. 11, No. S6, pp. 1–6, 2010.

28. Calderone, A., L. Licata and G. Cesareni, "VirusMentha: A New Resource for Virus-Host Protein Interactions", *Nucleic Acids Research*, Vol. 43, No. D1, pp. 588–592, 2014.

29. Girshick, R. B., J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", arXiv:abs/1311.2524, 2013.

30. Yang, F., K. Fan, D. Song and H. Lin, "Graph-based Prediction of Protein-protein Interactions with Attributed Signed Graph Embedding", *BMC Bioinformatics*, Vol. 21, No. 1, pp. 1–16, 2020.

31. Gillespie, J. J., A. R. Wattam, S. A. Cammer, J. L. Gabbard, M. P. Shukla, O. Dalay, T. Driscoll, D. Hix, S. P. Mane, C. Mao, E. K Nordberg, M. Scott, J. R Schulman, E. E Snyder, D. E Sullivan, C. Wang, A. Warren, K. P Williams, T. Xue, H. Seung Yoo, C. Zhang, Y. Zhang, R. Will, R. W Kenyon and B. W Sobral, "PATRIC: the Comprehensive Bacterial Bioinformatics Resource with a Focus on Human Pathogenic Species", *Infection and Immunity*, Vol. 79, No. 11, pp. 4286–4298, 2011.

32. Tekir, S. D., T. Çakır, A. S. Sayılırbaş, E. Çelik, S. Ozcan, I. Çevik, A. S. Ozcelik, A. Ozgur, F. E. Sevilgen and K. O. Ulgen, "PHISTO: Pathogen-Host Interaction Search Tool", *New Biotechnology*, Vol. 29, pp. 1357–1358, 2012.

33. Guirimand, T., S. Delmotte and V. Navratil, "VirHostNet 2.0: Surfing on the Web of Virus/Host Molecular Interactions Data", *Nucleic Acids Research*, Vol. 43, No. D1, pp. 583–587, 2014.

34. Qi, Y., O. Tastan, J. G. Carbonell, J. Klein-Seetharaman and J. Weston, "Semi-Supervised Multi-Task Learning for Predicting Interactions Between HIV-1 and Human Proteins", *Bioinformatics*, Vol. 26, No. 18, pp. i645–i652, 2010.

35. Hashemifar, S., B. Neyshabur, A. A. Khan and J. Xu, "Predicting Protein-Protein Interactions through Sequence-Based Deep Learning", *Bioinformatics*, Vol. 34, No. 17, pp. i802–i810, 2018.

36. Li, J. J. and Y. H. Chen, "Auto Covariance Combined with Artificial Neural Network for Predicting Protein-Protein Interactions", *Advanced Materials Research*, Vol. 765-767, pp. 1622–1624, 2013.

37. Guo, Y., L. Yu, Z. Wen and M. Li, "Using Support Vector Machine Combined with Auto Covariance to Predict Protein-Protein Interactions from Protein Sequences", *Nucleic Acids Research*, Vol. 36, No. 9, pp. 3025–3030, 2008.

38. You, Z.-H., L. Zhu, C.-H. Zheng, H.-J. Yu, S.-P. Deng and Z. Ji, "Prediction of Protein-Protein Interactions from Amino Acid Sequences Using a Novel Multi-Scale Continuous and Discontinuous Feature Set", *BMC Bioinformatics*, Vol. 15, No. Suppl 15, pp. 1–9, 2014.

39. You, Z.-H., K. C. Chan and P. Hu, "Predicting Protein-Protein Interactions from Primary Protein Sequences Using a Novel Multi-Scale Local Feature Representation Scheme and the Random Forest", *PLOS ONE*, Vol. 10, No. 5, 2015.

40. Yang, X., S. Yang, Q. Li, S. Wuchty and Z. Zhang, "Prediction of Human-Virus Protein-Protein Interactions through a Sequence Embedding-Based Machine Learning Method", *Computational and Structural Biotechnology Journal*, Vol. 18, pp. 153–161, 2020.

41. Ponti, E. M., I. Vulić and A. Korhonen, "Decoding Sentiment from Distributed Representations of Sentences", *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, pp. 22–22, Vancouver, Canada, August 2017.

42. Gonzalez-Lopez, F., J. A. Morales-Cordovilla, A. Villegas-Morcillo, A. M. Gomez and V. Sanchez, "End-to-End Prediction of Protein-Protein Interaction Based on Embedding and Recurrent Neural Networks", *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 2344–2350, Madrid, Spain, December 2018.

43. Du, X., S. Sun, C. Hu, Y. Yao, Y. Yan and Y. Zhang, "DeepPPI: Boosting Prediction of Protein-Protein Interactions with Deep Neural Networks", *Journal of Chemical Information and Modeling*, Vol. 57, No. 6, pp. 1499–1510, 2017.

44. Li, H., X.-J. Gong, H. Yu and C. Zhou, "Deep Neural Network Based Predictions of Protein Interactions Using Primary Sequences", *Molecules*, Vol. 23, No. 8, pp. 19–23, 2018.

45. Kimothi, D., A. Soni, P. Biyani and J. M. Hogan, "Distributed Representations for Biological Sequence Analysis", arXiv:abs/1608.05949, 2016.

46. Barman, R. K., S. Saha and S. Das, "Prediction of Interactions Between Viral and Host Proteins Using Supervised Machine Learning Methods", *PLoS ONE*, Vol. 9, No. 11, pp. 1–10, 2014.

47. Lanchantin, J., T. Weingarten, A. Sekhon, C. Miller and Y. Qi, "Transfer Learning for Predicting Virus-Host Protein Interactions for Novel Virus Sequences", *Proceedings of the 12th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pp. 1–10, Gainesville Florida, August 2021.

48. Hoehndorf, R., P. N. Schofield and G. V. Gkoutos, "PhenomeNET: A Whole-Phenome Approach to Disease Gene Discovery", *Nucleic Acids Research*, Vol. 39, No. 18, pp. 119–119, 2011.

APPENDIX A: Additional Tables

Table A.1. Hyperparameters for edge filtering Ebola.

Hyperparameter	**Cut-off Value**
PMI	Bigger than 13
TF-IDF	Biggest 250
Needleman-Wunsch	Bigger than 0.5

Table A.2. Graph details: Ebola dataset.

Nodes	**Count**
Human	2263
Virus	385
Word	4755
Edges	
Sequence - Contains - Word	497478
Human - Interacts - Virus	4545
Human - Not Interacts - Virus	4233
Human - Similar - Human	5121169
Virus - Similar - Virus	148225
Word - PMI - Word	22610025

Table A.3. Hyperparameters for edge filtering H1N1.

Hyperparameter	**Cut-off Value**
PMI	Bigger than 13
TF-IDF	Biggest 250
Needleman-Wunsch	Bigger than 0.5

Table A.4. Graph details: H1N1 dataset.

Nodes	**Count**
Human	2263
Virus	385
Word	4755
Edges	
Sequence - Contains - Word	497478
Human - Interacts - Virus	4545
Human - Not Interacts - Virus	4233
Human - Similar - Human	5121169
Virus - Similar - Virus	148225
Word - PMI - Word	22610025

Table A.5. Hyperparameter analysis on the graph model for Ebola dataset.

		ACC	**F1**	**AUC**	**MCC**
PMI	**>12**	0.865 +-0.015	0.872 +-0.016	0.941 +-0.006	0.736 +-0.033
	>13	0.869 +-0.009	0.869 +-0.012	0.943 +-0.003	0.743 +-0.017
	>14	0.868 +-0.008	0.879 +-0.008	0.915 +-0.016	0.748 +-0.018
TF-IDF	**max 50**	0.852 +-0.004	0.857 +-0.009	0.933 +-0.004	0.709 +-0.011
	max 250	0.865 +-0.015	0.872 +-0.016	0.941 +-0.006	0.736 +-0.033
	max 500	0.863 +-0.012	0.871 +-0.012	0.941 +-0.012	0.734 +-0.025
Needleman Wunsch	**>50**	0.865 +-0.015	0.872 +-0.016	0.941 +-0.006	0.736 +-0.033
	>75	0.835 +-0.014	0.831 +-0.017	0.920 +-0.014	0.672 +-0.028
	>90	0.846 +-0.024	0.846 +-0.027	0.922 +-0.017	0.696 +-0.052

Table A.6. Hyperparameter analysis on the graph model for H1N1 dataset.

		ACC	**F1**	**AUC**	**MCC**
PMI	**>12**	0.898 +-0.013	0.902 +-0.011	0.961 +-0.017	0.799 +-0.024
	>13	0.900 +-0.033	0.903 +-0.031	0.965 +-0.018	0.802 +-0.066
	>14	0.900 +-0.027	0.898 +-0.027	0.968 +-0.008	0.800 +-0.055
TF-IDF	**max 50**	0.916 +-0.016	0.916 +-0.017	0.971 +-0.009	0.832 +-0.033
	max 250	0.900 +-0.033	0.903 +-0.031	0.965 +-0.018	0.802 +-0.066
	max 500	0.906 +-0.015	0.906 +-0.017	0.974 +-0.008	0.814 +-0.032
Needleman Wunsch	**>50**	0.900 +-0.033	0.903 +-0.031	0.965 +-0.018	0.802 +-0.066
	>75	0.910 +-0.025	0.911 +-0.026	0.974 +-0.003	0.823 +-0.052
	>90	0.872 +-0.021	0.865 +-0.026	0.964 +-0.004	0.749 +-0.040