Indraprastha Institute of Information technology Delhi

and

Queensland University of technology

Doctoral Thesis

Learning Representations for Molecular Sequences

Author:
Dhananjay Kimothi

Principal Supervisors:
Dr. Pravesh Biyani and Dr. James M Hogan

Associate Supervisor:
Dr. Wayne Kelly

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

October 4, 2021

Certificate (IIITD)

This is to certify that the thesis titled, "Learning Representations for Molecular Sequences" being submitted by Dhananjay Kimothi to the Indraprastha Institute of Information Technology Delhi and Queensland University of Technology, Australia, for the award of the degree of Doctor of Philosophy, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

Dr. Pravesh Biyani
Department of ECE
IIITD, India

Dr. James M Hogan
School of Computer Science
QUT, Australia

Declaration (QUT)

The work contained in this joint thesis undertaken between Indraprastha Institute of Information Technology Delhi and Queensland University of Technology has not been previously submitted to meet requirements for an award at these or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed:  Date: 08/02/2021

विद्या ददाति विनय् विनयाद्याति पात्रताम् ।
पात्रत्वाद्धनमाप्नोति धनाद्धर्मम् तत : सुखम् ।
vidyA dadAti vinayaM, vinayAdyAti pAtratAM |
pAtratvAddhanamApnoti, dhanAddharmaM tataH sukhaM || 5 ||
(Hitopadesha)

(true/complete) knowledge gives discipline, from discipline comes worthiness, from worthiness one gets wealth, from wealth (one does) good deeds, from that (comes) joy.

Abstract

Dhananjay Kimothi

Learning Representations for Molecular Sequences

Sequence comparison is a vital step in bioinformatics tasks such as annotation of molecules, phylogeny construction, and sequence retrieval. Methods for sequence comparison are broadly divided into alignment-based and alignment-free approaches. Alignment-free methods offer some computational advantages – with some loss of sensitivity - and usually rely on high dimensional vector representations based on the bag of words model. This representation excludes contextual information from the sequences. Recent work on representation in Natural Language Processing (NLP) has gained wide popularity across a number of domains. These methods essentially work on the "distributional hypothesis" – that words that occur together frequently have some semantic relationship - and provide a way to generate low dimensional distributed representations of words or sentences while considering contextual information.

Motivated by these works in NLP, in this thesis, we aim to address the limitations of alignment and alignment-free methods by developing representation learning methods for bioinformatics tasks. It is notable that for many problems in bioinformatics, the available metadata also plays a role in biological inference. Representation learning frameworks allow metadata to be accommodated along with contextual information in the process of generating sequence representations. More specifically, this research aims to develop scalable, computationally efficient and competitive alignment-free solutions for bioinformatics problems such as protein classification, retrieval, and protein-protein interaction predictions.

The main contributions of this thesis are as follows:

- `Seq2Vec` – a new unsupervised framework for learning useful low dimensional representations of molecular sequences.
- `SuperVec` and `SuperVecX` – novel approaches for fusing meta and sequence information to generate improved embeddings of molecular sequences.
- `H-SuperVec(X)` – a hierarchical algorithm utilising learned representations for sequence comparisons, achieving performance comparable to alignment-based approaches at substantially lower computational cost.
- A hybrid system that utilises these alignment-free approaches as a rapid pre-processing filter to reduce the candidate set for an alignment-based algorithm, yielding a substantial speed up in the overall process.
- Demonstrated utility of these representation-learning based approaches for a variety of bioinformatics problems, e.g., protein family prediction, protein-protein interactions and homologous sequence retrieval.

The representation learning and task-specific approaches proposed in this thesis are generic and can be adapted for similar problems within bioinformatics and other domains.

Acknowledgements

It is that time when I would like to take a pause and trace back this fascinating journey of my PhD research. When writing this piece and thinking back, I realize, I have covered a lot of intellectual and academic space from where I initially started. I would be lying if I do not acknowledge that this whole process did take an emotional, physical, and psychological toll, but in hindsight I feel that is part of the journey. I faced multiple ups and downs in this journey, but it never stopped fascinating and exciting me. I did wander, but I knew in my heart that I am not lost, and I will reach the end of the journey, and here I am. It would not have been possible for me to travel this far, if it was not for many people who helped me at different stages of this journey.

First, I would like to acknowledge the support and guidance I received from my supervisors – Dr. Pravesh Biyani and Dr. James M. Hogan. I am thankful to them for forging this collaboration and facilitate me to work in the interdisciplinary domain of bioinformatics and machine learning. This collaboration enriched my experience not only academically but at a personal level as well. In these years, I got an opportunity to experience two different education systems and work in a collaborative setup. During this course, my stay in Brisbane, Australia, helped broaden my view of the world, meet people from different nationalities, and come close to other cultures.

Further, I would like to thank - Dr. Wayne Kelly (my associate supervisor at QUT) who helped me write optimized codes for the models proposed in this thesis. Dr. Akshay Soni (currently working with Microsoft) for introducing me to the representational learning research and hand-holding me in initial stages of my research. Dr. Ganesh, Dr. Sriram and Dr. Saket (my panel member from IIIT-Delhi) for providing me feedback on my progress and being available for discussions and guidance whenever required. Professor David Lovel, Dr. Tony Parker and Dr. Dimitri Parrin (my QUT examination panel)

for providing detailed feedback on my thesis draft. It is of utmost importance to have family and friends support to maintain focus and balance while working with tight schedules, stress at times, and deadlines. I feel I am lucky in this regard. During these years, some colleague who later became good friends contributed indirectly towards completing this thesis. Annupriya was always available for technical discussions and chit-chat. She influenced me with her work ethics, clarity of thoughts, and sincerity towards the work. Lokender, Ambuj and Parag remained supportive for resolving any issues however small it may be. In the last couple of years, Anand and Charul kept me motivated to keep coming to Lab; late evening chats with them acted as a stress buster.

I would like to acknowledge and appreciate support I got from my friend Saarim, for the patience with which he listened to my rant whenever I was not happy (which happened to be on several occasions) with the way things were going over these years. I have no words to acknowledge the love and support I got from my wife, Monica. She remained calm, tolerant, and patient while I worked on finishing this thesis. Her ability to adapt to different situations made it easy for me to balance my commitments towards family and PhD research. I entirely relied on her wherever the family required my presence. She even contributed in proof-reading some part of the thesis as well.

Lastly, nothing would have been possible without my parents' blessings and love and support of my entire family.

Contents

List of Figures

List of Tables

List of Abbreviations

RepL	**R**epresentation **L**earning
BLAST	**B**asic **L**ocal **A**lignment **S**earch **T**ool
AF	**A**lignment **F**ree
PPI	**P**rotein **P**rotein Interaction
GO	**G**ene **O**ntology
DNA	**D**eoxyribonucleic **A**cid
RNA	**R**ibonucleic **A**cid
NLP	**N**atural **L**anguage **P**rocessing
ML	**M**achine **L**earning
NMR	**N**uclear **M**agnetic **R**esonance
BLOSUM	**B**LOcks **S**UbstitutionMatrix
CBOW	**C**ontinious **B**ag of **W**ords
DAG	**D**irected **A**cyclic **G**raph
RF	**R**andom **F**orest
SVM	**S**upport **V**ector **M**achine
BOW	**B**ag **O**f **W**ords

Dedicated to Papa and Ma:
Dr. Rajendra Prasad Kimothi
and Smt. Suman Kimothi

1 Introduction

Overview

Biological sequence analysis is a core research area in bioinformatics with applications including gene annotations, function and structure predictions, homologous sequence retrieval and phylogeny construction [1]. So far, sequence analysis research has been dominated by the alignment-based methods [2] that rely on computing sequence similarity by aligning them. These approaches are accurate but are memory consuming and time consuming [3]. The alternative approaches (also known as alignment-free approaches [2]) provide computational gain with some loss of sensitivity. The current alignment-free methods usually rely on high dimensional sequence representations. These representations are naive, and hence there remains scope for improvement. In this thesis, we focus on developing approaches that can generate low dimensional yet useful representations of sequences and hence address some of the limitations of alignment-based and alignment-free methods.

In this chapter, we provide a brief introduction to sequence analysis and sequence comparison techniques. Later in the chapter, we underline the motivating factors leading to the research pursued in this thesis. Lastly, we briefly explain the objectives and contributions of this thesis. In the end, we provide an overview of follow up chapters.

1.1 Introduction to Sequence Analysis

Biological macromolecules (Biomolecules) such as nucleic acids - DNA (deoxyribonucleic acid), RNA (ribonucleic acid) and proteins are key molecules present in cells. Their direct or indirect involvement in cellular processes makes it essential to understand their properties, function and higher-level

structures. The most reliable way to do so is through experimental approaches, e.g., mass spectrometry [4], yeast two-hybrid method [5]. These approaches are costly and time-consuming and hence pose challenges when scalability is desired. Considering the limitation of experimental techniques, researchers focused on developing complementary in-silico approaches. These approaches focus on processing the primary structure of biomolecules for inferring biological information.

The primary structure (often referred just as sequence) of a biomolecule is its low-level representation where the constituent molecules (each represented as a character of restricted alphabet) are linearly arranged. For example, a DNA

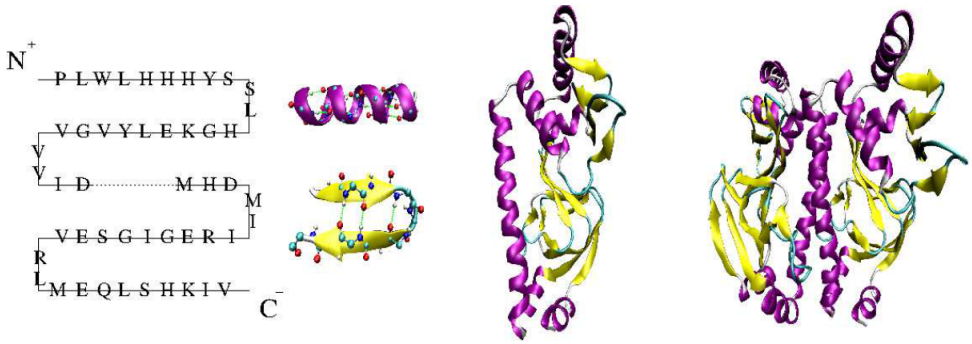


Figure 1.1: ***Levels of protein structure : Primary, Secondary, Tertiary and Quaternary structures.***

sequence "ACTCGCTACTGC" is composed of characters drawn from the set of A, T, G, C which denotes nucleic acid molecules - adenine, thymine, guanine, cytosine respectively. Similarly, protein sequences draw characters from a set of 20 amino acids. This linear arrangement (primary structure) of characters is an abstraction of the actual complex 3D structure of the DNA/Protein or RNA molecule but retains vital information which can be utilized to infer the structure and function of the molecule or host organism. Fig. 1.1 shows the primary and other higher-level structure of a protein.

Analyzing macromolecule (DNA, Protein or RNA) sequences by subjecting them to different methods for inferring structure, function or evolution is a core research area in bioinformatics. These methods are collectively termed as *sequence analysis methods*. Comparing sequences and quantifying their similarity is a fundamental step to most of these methods. Some of the problems

that are addressed using *sequence comparison methods* include phylogeny construction, homology detection, gene annotation, protein-protein interaction prediction and others.

Algorithms for comparing sequences are broadly classified into two categories, namely, alignment-based and alignment-free algorithms. In the sections below, we briefly introduce a range of sequence comparison methods.

1.2 Sequence Comparison methods

Sequence comparison methods can be broadly classified as either alignment-based and alignment-free methods. A brief introduction to each of the categories is given below.

1.2.1 Alignment-based methods

These methods mainly concern positioning the constituent alphabets of characters with one another to identify the regions of similarity. Such similarities can be spread across the sequences, or concentrated only in some local regions. Needleman et al. [6] and Smith et al. [7] were among the first to propose algorithms for finding global and local similarities respectively. These algorithms utilise dynamic programming and a scoring matrix to find and quantify the similarity between sequences. The scoring matrix defines the scores for match, indel (insertion, deletion) and substitution; these scores are considered to be directly related to the biological events of conservation and mutation. In the computation of the similarity score, conservation is rewarded with a higher score, whereas indel and substitution are penalised.

Over the years, many successful sequence analysis tools were developed that use alignment as a core component for sequence comparison. Some of the examples of such tools include: (i) sequence search tools such as BLAST and FASTA [8], (ii) multiple sequence alignment tools like ClustalW [9], Muscle [10], MAFFT [11], (iii) sequences profile search program - PSI-BLAST [12], HMMER [13] and (iv) whole genome aligners like progressive Mauve [14], BLASTZ [15] and TBA [16]. While the demonstrated history of sequence alignment(s) as an effective method for sequence comparison makes them a

reliable option for many bioinformatics applications, there are some inherent downsides associated. We consider these below.

Alignment-based techniques assume continuity (i.e. homologous sequences contain the linearly arranged conserved regions) which fails in some cases where the related biomolecules have divergent sequences. The variation in viral genomes because of high mutation rates and events like gene duplication and horizontal gene transfer provide good examples of the violation of the assumption of continuity. Also, the performance of alignment-based techniques deteriorates with decreasing sequence identity, e.g. gene regulatory sequences and distantly related proteins. Another important limitation is the time complexity and memory requirement by all alignment-based techniques. The complexity of aligning two sequences of length n using native algorithms (Needleman Wunsch [6] and SmithWaterman) [7]) is $\mathcal{O}(n^2)$[1] . More advanced tools like BLAST use heuristics to optimize speed, which affects their ability to identify the optimal highest-scoring alignment, which in turn affects the quality of downstream tasks like phylogeny reconstruction. Algorithms that rely on the alignment of multiple sequences provide improvement in accuracy as well sensitivity as compared to the pairwise alignment-based algorithm. But, finding an accurate Multiple Sequence Alignments (MSA) is an NP-hard problem[2] and can't be solved in a realistic time frame. Heuristic alternatives are effective but may limit the quality of the alignment produced.

1.2.2 Alignment-free methods

The limitations of alignment-based approaches for certain bioinformatics tasks has motivated researchers to develop the alternatives collectively known as

[1]For comparing two sequences of length $n = 100$, an algorithm with $\mathcal{O}(n^2)$ time complexity would require 10000 computations, for a 10000 length sequence, it requires 10^8 (hundred million computations) which may require minutes to hours to compute on a standard desktop. These computations become more challenging for problems like - 'sequence retrieval' 'where millions of pairs of sequences are required to be compared.

[2]The NP hard problems are a set of problems that cannot be solved in polynomial time; the computational effort required for these problems increases exponentially with the input size. They are said to have exponential computational complexity. A typical presented example is the Subset Sum problem as described in [17].

alignment-free methods [2]. *Alignment-free methods* is an umbrella term collectively used for sequence analysis methods that do not rely on the alignment of sequences.

The absence of a formal alignment within these algorithms for sequence comparison makes them computationally less expensive when compared to alignment-based approaches. Another advantage favouring these approaches is that they are not built on the evolutionary assumptions prevalent in alignment based approaches and hence are more resistant to shuffling and recombination events.

The essential component of an alignment-free (AF) method is to map and use sequences in a different domain. Most of the time this mapping is from a sequence to a point in R^n space, but other approaches such as graphical representation, iterated maps and matching words are also explored in some of the literature. An essential intermediate step in representing sequences as vectors is to tokenize the sequence into smaller sub-sequences known as *k-mers* or words. These *k-mers* are generated by moving a window of size k across the sequence. Statistics from the list of *k-mers* are then used as an alternative representation (feature vector) for a sequence. For example, counting *k-mers* - or a group of N *k-mers* - gives rise to 1-gram or N-gram representations respectively. Here we note that the 1-gram is same as the bag-of-words representation. Although such representations are simple to construct, they inherit two important limitations: the size of vector and limited context in the representation.

For biological sequences (e.g., protein, DNA or RNA) of alphabet size $|A|$, the length of 1-gram vector ($|A|^k$) increases exponentially with increasing window size k. Note that no contextual information used to construct 1-gram vectors. Although N-gram ($N = 2, 3, ..$) representations include the context in a limited fashion, introducing additional context (by choosing large N) dramatically increases the vector dimension. For window size k, the N-gram vector length is given as $|A|^{Nk}$. The rise in length of 1-gram/N-gram become severe in case of proteins ($|A| = 20$) as compared to DNA/RNA ($|A| = 4$). For a protein sequence where the 1-gram vector size is 20^k, for small rise of k, the dimension is still very large. The size of N-gram representation of protein

sequence, with a choice of $k = 3$ and $N = 2$ (a very limited context), is already $20^6 = 64,000,000$.

The other methods for feature vector construction for sequences utilise physicochemical properties of the constituent letters of a sequence. Such feature construction greatly depends on the ingenuity of the researcher in selecting properties relevant to the problem. This is not necessarily straightforward and does not generalises readily to other problems.

Having introduced alignment-based and alignment-free methods, considered their applications and noted their limitations, we summarise the motivating factors that lead to the research work pursued in this thesis.

- Alignment-based approaches present computational challenges for large scale sequence comparisons, which has become more critical with the exponential rise in sequences availability. Alignment-free approaches may provide computational advantages with some loss of sensitivity if we have methods that can give low dimensional yet useful representations of the sequences.
- Many of the current alignment-free approaches rely on constructing feature vectors for sequences using handcrafted features, which requires ingenuity and domain knowledge. Handcrafted features may miss some of the essential attributes which may otherwise be necessary for the task at hand. It motivates us to work on entirely data-driven approaches that don't rely on human interference and would be able to learn abstract features which may otherwise be missed.
- The alternative to handcrafted features for representing sequences is the bag-of-words based feature vector. Such feature vectors are naïve and may not capture all relevant information from the sequences. There remains scope to work on approaches that can capture the patterns present in and across the sequences to provide better sequence representations.
- To generate sequence representations most of the available methods only rely on the information present in the sequences. However, for some bioinformatics problems - such as homologous sequence retrieval, sequence-based inference might not be sufficient. In such cases using meta-information (such as functional or structural annotation) along

with sequence information may provide improved representations. Current alignment-free methods do not facilitate the use of meta and sequence information together for generating better sequence representations.

- The resemblance of sequences to text data allows us to explore the ideas used in representation learning research from text processing domains in the bioinformatics space. Their success in the text processing domain motivates us to explore similar approaches for biological sequences. Developing such approaches in the bioinformatics space may enable us to generate generic or task-specific, low dimensional yet effective representations of the sequences suitable for a range of bioinformatics problems.

In the section below, we briefly introduce the research direction pursued in this thesis to learn low dimensional representations of sequences.

1.3 Learning Representations for sequences

Growth in the availability of sequences in the last few years makes it possible to consider new approaches that may capture sequences' patterns in low dimensional representations. Along with the improvement in sequencing technologies, bioinformatics research has also witnessed a rapid growth in the availability of meta-information (e.g., functional, structural annotations) in the last few years. Availability of a sufficient amount of data opens an opportunity to apply deep learning or representation learning approaches that heavily rely on an adequate amount of data for training, in the bioinformatics domain, which was not possible before. Such approaches are expected to yield improved low dimensional sequence representations compared to more obvious high dimensional N-gram representations.

One of the possible directions in this context is to explore recently popularized distributed representation learning methods from the text processing domain. The work of Mikolov et al. [18] on word-embeddings and Le et al. [19] on sentence embeddings has gained considerable attention through the success of these methods in capturing contextual information as well as

generating low dimensional representations of text. These papers have motivated many other studies in the community and across domains. These works have inspired the work in this thesis in the bioinformatics domain.

In the section below, the research objectives of this thesis are summarized.

1.4 Research Objectives

As discussed above, both alignment and alignment-free methods have their limitations. With these issues in mind, the main aim of this research is to propose fast and competitive alignment-free methods, which involve learning generic or task-specific representations for biological sequences. The proposed methods also address some of the computational and dimensionality issues of existing techniques. Specifically, the objectives of this work are as follows:

- **Propose and evaluate an unsupervised method to learn low dimensional representation for biological sequences.**
- **Propose and evaluate a method that can utilize available meta information like functional annotation to improve the sequence representations.**
- **Propose and evaluate a method that provides faster alternative for homologous sequence retrieval problems.**
- **Demonstrate the utility of learned sequence representation for varied downstream bioinformatics tasks.**

In the following section, we briefly discuss the major contributions of this thesis.

1.5 Contributions

This thesis explores the utility of representation learning approaches for bioinformatics applications. We propose unsupervised and supervised methods to learn a lower dimensional vector representation (also known as an embedding) for protein molecules. The unsupervised approach captures the

contextual information present in the sequences to generate corresponding n-dimensional feature vectors. The supervised approach enriches the sequence embeddings with available meta-information.

Sequence embeddings generated through these approaches can be used to build methods for many downstream classification, clustering or retrieval tasks. This thesis demonstrates the direct application of these embeddings for protein family prediction, homologous sequence retrieval and various protein classification tasks. These includes toxin prediction, subcellular location prediction and enzyme prediction. For the homologous sequence retrieval task a generic framework is proposed that is built on the supervised representation learning methods.

Apart from the bioinformatics applications in which the inference is made only about individual proteins, this thesis also explores the use of learned sequence representations on the paired input problem, specifically for protein-protein interaction prediction task.

The main contributions are described in detail below.

1.5.1 Unsupervised Representation learning for Molecular Sequences: Seq2Vec

- Our contribution in this work is two fold. First, we propose an unsupervised method `Seq2Vec` to generate a low dimensional representation of molecular sequences. Using this method, the contextual information (i.e. the co-occurrence of *k-mers*) present in the sequences is captured within the feature vector.
- Secondly, we use `Seq2Vec` to generate embeddings for protein sequences and demonstrate high accuracy on a protein family classification task. Such high accuracy demonstrates that contextual information in sequences is an important feature that can capture biologically relevant information.

  Details of this work are provided in Chapter 3.

1.5.2 Supervised representation learning for Molecular sequences: SuperVec(X)

- We present a supervised approach – `SuperVec` – to learn embeddings for biological sequences. `SuperVec` provides the flexibility to utilize meta-information (like class labels, e.g. protein family annotations) along with the contextual information present in the sequences to generate sequence embeddings. It is important to note here that only the training of the proposed model is supervised and the generation of embeddings remains an unsupervised procedure.
- We present `SuperVecX` – an alternative supervised embedding technique for generating biological sequence embeddings. The `SuperVecX` architecture is motivated from a recently popularized document classification technique called `fastText`.
- We show that embeddings generated using a supervised approach are superior to the embeddings generated through unsupervised methods such as `BioVec` [20] and `Seq2Vec`, capturing both class and sequence information. Better clustering of homologous sequences validates our hypothesis that supervision would improve the embeddings.

  Details are provided in chapter 4

1.5.3 Homologous Sequence Retrieval Problem

The ability of `SuperVec(X)` to produce low dimensional representations that are improvement over unsupervised approaches makes it a useful tool for large scale sequence comparison problems. One such task is *homologous sequence retrieval*. This task pertains to retrieving sequences from a database that are homologous to a given query sequence. To address the homologous sequence retrieval problem we present two approaches:

An Hierarchical approach

- When trained with different set of sequences, `SuperVec(X)` can produce diverse set of representations for sequences which enables us to compute more than one distance between the sequences in a vector space. A

weighted combination of these distances is expected to provide a better estimate of similarity between a pair of sequences, this forms the basis of our proposed hierarchical approach. We call this approach either `H-SuperVec` or `H-SuperVecX` depending upon the underlying embedding method – respectively `SuperVec` or `SuperVecX`.

- We show that our approaches provide a faster alternative to alignment-based methods like BLAST and MMseqs2 for the sequence retrieval task, providing an order of magnitude speedup in querying time.

An Hybrid Approach

- We show that our approaches can be used as a pre-processing filter prior to the application of a slower, high precision method for sequence retrieval tasks over a smaller dataset – allowing these results to be obtained far more rapidly.

Details of these approaches are provided in chapter 5

1.5.4 Learned sequence representations and their applications

Learning useful sequence representations is an evolving domain in bioinformatics. Here we demonstrate the effectiveness of these methods on a range of bioinformatics tasks. The successful application of these methods also opens up the possibility of more advanced methods to cater for specific bioinformatics tasks. The applications in which we successfully demonstrate the use of learned representations include:

- Protein family classification (Chapter 3)
- Cellular localization of proteins (Chapter 4)
- Enzyme Classification (Chapter 4)
- Toxin Prediction (Chapter 4)
- Homologous sequence retrieval (Chapter 5)
- Protein-Protein Interaction Prediction (Chapter 7, Chapter 8)

We study the protein-protein interaction prediction task in detail; the contributions related to this work are summarized below.

1.5.5 Protein-Protein interaction prediction problem

PPI prediction is a paired-input problem where the task is to determine the possible relationship between two proteins. While there are different ways to approach PPI problem, we limit our focus on two data sources - sequences and gene ontology terms. Our contributions are summarized below.

PPI prediction using sequences

- We use sequence embedding based protein pair representations for the protein-protein interaction prediction task. The representation of each protein pair consists of the embeddings of its constituent protein sequences. We demonstrate that such representations yield improved results when compared to most of the available physico-chemical property based protein-pair representations.
- We introduce a new sequence representation learning model - `SuperVecNW` which utilizes known binary relationships (interaction/non-interaction) along with contextual information to learn sequence embeddings. We use these embeddings to construct representations for protein pairs that are further used for PPI predictions.
- We demonstrate that our approach achieves comparable performance to that of much more complex end-to-end deep learning approaches on the PPI prediction task.

  Details of this work are provided in chapter 7.

PPI prediction using GO terms

- We propose an approach to learn low dimensional representations for GO terms using the existing relationship among GO terms (obtained from GO-DAG [21]) of protein pairs and known interactions.

- With experiments, we show that protein pair representations obtained by combining GO term embeddings provide improved results over sequence based inference. Note that the limited availability of GO annotations inhibit such methods being used for all protein pairs. Possibly, an integrated approach that considers both sequences and GO terms can address their individual limitations and provide a better framework to predict PPIs.

  Details of these experiments are provided in chapter 8

1.6 Publications arising from this research

The work presented in this thesis has also appeared in the following accepted and submitted publications.

- **Dhananjay Kimothi**, Pravesh Biyani, James M. Hogan, Akshay Soni, and Wayne Kelly. "Learning supervised embeddings for large scale sequence comparisons." PloS one 15, no. 3 (2020): e0216636. [citations: 3]
- **Dhananjay Kimothi**, Pravesh Biyani, and James M. Hogan. "Sequence representations and their utility for predicting protein-protein interactions." bioRxiv (2020).
- **Dhananjay Kimothi**, Pravesh Biyani, James M. Hogan "GO label representations and their ability to predict PPI" (*Under Prepration*)
- **Dhananjay Kimothi**, Ankita Shukla, Pravesh Biyani, Saket Anand, and James M. Hogan. "Metric learning on biological sequence embeddings." In 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 1-5. IEEE, 2017. [citations: 3]
- **Dhananjay Kimothi**, Akshay Soni, Pravesh Biyani, and James M. Hogan. "Distributed representations for biological sequence analysis." arXiv preprint arXiv:1608.05949, presented at DTMBIO-2016 (Workshop at CIKM 2016) [citations: 36]

1.7 Structure of the Thesis

This thesis is organized as follows.

Considering the interdisciplinary nature of the research pursued in this thesis, we cover background and literature related to sequence comparison methods and representation learning techniques in chapter 2. Initially, we consider alignment and alignment-free sequence comparison techniques and their advantages and disadvantages. This is followed by a discussion on representation learning techniques and a detailed description of the word embedding model. Further, we summarise recent developments regarding the usage of representation learning based approaches in the bioinformatics domain.

Having set the background for the thesis in chapter 2, chapter 3 introduces our proposed unsupervised representation learning technique `Seq2Vec`. `Seq2Vec` processes sequences to directly generate useful embeddings. We apply these embeddings for the protein family classification task and show improved results compared to the existing benchmark embedding technique `BioVec`. We also analyse and compare the distribution of these embeddings in the vector space and show that `Seq2Vec` generated embeddings provide better clusters of sequences that belong to the same protein family when compared with `BioVec`.

Based on the insights gained from the work discussed in chapter 3, in chapter 4, we introduce two new representation learning techniques: `SuperVec` and `SuperVecX`. These models are capable of capturing meta-information along with the sequence information. We show that though the embeddings generated through `SuperVec` and `SuperVecX` are of low dimension, they still provide comparable or better results on various protein classification tasks when compared to the benchmarked feature construction approaches.

In chapter 5, we focus on the homologous sequence retrieval task. First, we demonstrate that the embeddings generated through `SuperVec(X)` models give better inter class separation and intra-class locality in the vector space when compared to the unsupervised techniques like `BioVec` and `Seq2Vec`. We then demonstrate the utility of `SuperVec(X)` generated embeddings on the sequence retrieval task, while noting that the accuracy does not reach the same level as BLAST. To help address this deficit we propose a hierarchical

approach build on SuperVec(X) specifically for "sequence retrieval" and call it H-SuperVec(X) and a hybrid approach - H-SuperVec(X)+BLAST. We finally achieve results comparable to alignment-based techniques while still providing an order of magnitude reduction in querying time.

Having introduced methods for learning representations of biological sequences and their applications for various bioinformatics tasks, we extend these ideas to the protein-protein interaction prediction (PPI) problem. PPI is a paired-input problem, where the task is two determine the relationship (interaction or no-interaction) between two proteins. In the next set of chapters, we focus on PPI prediction problem.

In chapter 6, we provide the background of the PPI prediction problem. We discuss different experimental and computational approaches and their limitations proposed for predicting PPIs. We review the literature related to sequence and GO-term based methods in detail. In the end, we discuss the issues related to the evaluation of PPI prediction methods that may give the wrong impression of their capability to generalise at the population level.

In chapter 7, we explore the utility of sequence representations for the PPI prediction task. We present a PPI prediction framework which uses learned sequence embeddings to represent a protein pair. To generate sequence embeddings, we use the proposed representation learning techniques described in chapter 3 and 4. We also introduce a new Representation Learning framework SuperVecNW that uses some of the information from the PPI network and sequences to generate sequence embeddings. We demonstrate that using our approach, better PPI predictions can be obtained without relying on the physico-chemical properties of amino-acids.

In chapter 8, we explore the utility of GO term representations for the PPI prediction task. We present a framework that utilises the existing relationship among GO terms and their participation in known interactions to generate their embeddings. These embeddings are combined to represent protein pairs. We demonstrate that GO terms based methods provide better predictions than those relying only on sequences. On the downside, limited availability of GO terms inhibit their applications for predicting possible interactions for all protein pairs. Working on a joint approach that uses both

sequences and GO terms might yield improved results.

Chapter 9 concludes this thesis by summarizing the contributions and proposing several possible future research directions.

2 Literature Review

Overview

In the previous chapter, we gave an overarching view of the thesis - providing an introduction to the sequence analysis techniques, stating the motivating factors that lead to this research and summarising our contributions. This chapter elaborates on the topics that are briefly introduced in the previous chapter. Since our work lies in the intersection of sequence analysis and representation learning research, we consider relevant literature from both of these areas. We first review research works related to alignment-based and alignment-free methods, noting their advantages and limitations. After that, we provide an overview of representation learning research and record their usages in several domains, including text, audio and images. Lastly, we review recent publications related to the representation learning research in the bioinformatics domain. The organisation of this chapter is given below.

We begin by reviewing literature concerning sequence alignment algorithms in section 2.1. It includes a detailed description of steps (seeding, extension and evaluation) followed for sequence comparisons in BLAST (section 2.1.1), and details of a recent sequence comparison tool, MMseqs2 (section 2.1.2). Both of these tools are used to benchmark the performance of our proposed approaches for *homologous sequence retrieval task* (discussed in chapter 5).

Section 2.2 is devoted to the alignment-based methods. It provides review related to the word frequency based methods (section 2.2.1) and information theory-based methods (section 2.2.2), in the end of this section limitations of the current alignment-based methods are summarised.

We note that limitations of alignment-based and current alignment-free methods can be addressed to some extent through Representation Learning (RepL)

based solutions. Section 2.3 provides a review of RepL approaches, noting their usage in other domains. Later in section 2.3.1, we provide details of a naive feed-forward neural network - that forms basis for two widely popularised word/document embedding approaches - `Word2Vec/Doc2Vec`. These embedding approaches inspire the models proposed in this thesis for learning representations of biological sequences. Details of `Word2Vec` and `Doc2Vec` is provided in section 2.3.2 and section 2.3.3 respectively.

Lastly, we provide review of some of the recent representation learning works in the bioinformatics domain (section 2.3.4) and conclude the chapter (section 2.3.5).

2.1 Sequence Alignment Algorithms

Needleman et al. [6] introduced the earliest work on alignment of a pair of biological sequences; this paper focused on the global alignment problem. Global alignment works on the principle of dynamic programming wherein a given sequence alignment task is fragmented into smaller tasks. The Needleman Wunsch alignment algorithm provides not only the optimal alignment of sequences but also quantifies the similarity of sequences by summing the alignment score of individual sequence elements. This algorithm is effective when used to infer functional similarity of two similar sequences of the same length but fails when the sequences are divergent and still perform the same function. To address this issue Smith et al. [7] extended the use of dynamic programming and proposed local alignment algorithm which outputs the sub-sequence shared by two sequences that yield the highest pairwise score. Evolutionarily related sequences share similar sub-sequences and hence preserve local similarity. Local alignment algorithms therefore are often more sensitive in detecting the evolutionary relationship of two sequences than a global alignment algorithm.

While local alignment algorithms are highly sensitive and performs well when employed for inferring sequence similarity, its quadratic complexity makes it non-viable for applications which involve large-scale sequence comparisons like database search. In a database search application, a query is compared

against all sequences in the database; the searched sequence is known as query whereas database sequences are called as subjects or targets.

Rapid growth in the availability of sequences and the poor scaling behaviour of dynamic programming based algorithm has lead to the development of faster methods. The earliest work used heuristics along with the alignment of sequences to get speedy execution at the cost of sensitivity. Collectively such methods are called heuristic search methods. FASTP [22] was the first such method; this algorithm replaces the dynamic programming based alignment algorithms with a step-wise approximate algorithm based on the diagonal. The main steps in FASTP involve identifying short identical substrings and then combining them to produce local alignment candidates. These candidates are then re-scored using the pairwise similarity score taken from a substitution matrix like PAM250 [23], to give the highest scoring candidate as a result. FASTA [24] extended the approach adopted in FASTP but differs in the selection of local alignment candidates, which are increased from 5 to 10. Also, unlike FASTP which returns the un-gapped alignment, FASTA achieves greater sensitivity by combining the high score regions to give the gapped alignments. In a database search, the highest ranking sequences obtained from FASTA are subjected to a form of Needleman Wunsch algorithm to produce global alignments. By far the most widely used approach for sequence comparison is BLAST (Basic Local Alignment Search Tool) which implements a heuristic which outweighs the FASTP/FASTA method in terms of execution speed at a tolerable cost to sensitivity. A detailed description of the BLAST algorithm is covered in the section below.

2.1.1 BLAST Algorithm

Similar to FASTA, BLAST first reduces the search space by finding seeds - short exact or approximate word [1] matches. The search space for two sequences can be visualized as two dimensional space containing ungapped and gapped alignments (refer Fig. 2.1).

The word matches identified in the search space are used as a seed for local alignment. Basically, BLAST adopts a "seed and grow" approach which

[1] words which are close; closeness here means the words which have match score (computed using the substitution matrix like BLOSUM [25]) within a threshold

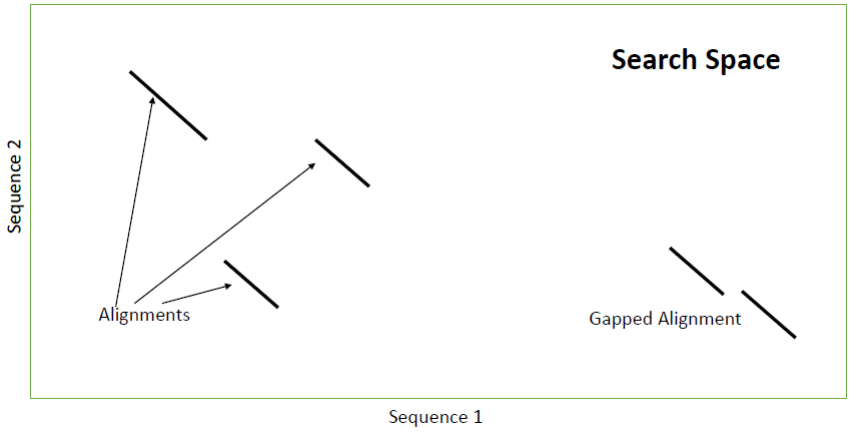


Figure 2.1: ***An example of search space.***

means first the seed is identified and then extended using a dynamic programming algorithm till a pre-defined threshold is attained. In summary, BLAST adopts three level of refinement - (i) seeding (ii) extension and (iii) evaluation to produce statistically relevant alignments. The details of these steps are discussed below.

Seeding

This step is based on the assumption that significant alignments share common words (*k-mers*). Location of the word matches (seed) is often termed as "word hits". For example, for sequence1 = MTASVKMTNLTASKA and sequence2 = MTASVKLANLTASVN the "word hits" are indicated with red color in Fig. 2.2. Often the significant alignments do not contain the exact word matches and to address this issue BLAST uses the concept of neighbourhood. The neighbourhood of a word (w) is considered to be all those words whose alignment score is higher than a certain threshold (T). A choice of T is made based on the required balance between sensitivity and speed. Higher values of T reduce the neighbourhood of a word and hence the word hits. A reduced neighbourhood, in consequence, reduces the search space and thus improves the speed. The improvement in speed comes at the risk of losing some alignments and hence the sensitivity of the algorithm.

Each seed computed at this stage acts as the starting of a candidate for a statistically significant alignment following the extension step.

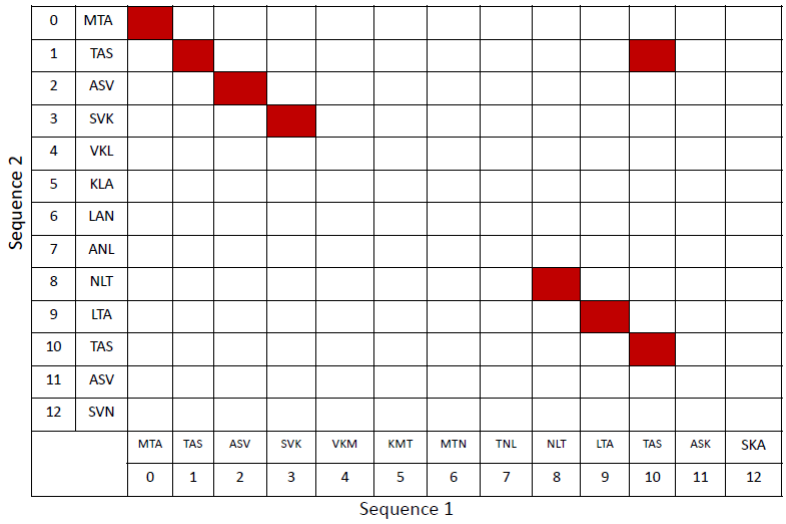


Figure 2.2: ***Seeding: Figure demonstrate the word hits of sequence1 = MTASVKMTNLTASKA and sequence2 = MTASVKLANLTASVN***

Extension

In this step, the seeds are extended in both directions (except for the seeds in corners) to produce alignments. Along with the extension, the alignment score and the drop in alignment score (d) are tracked. The alignment extension is stopped when the drop score hits a predetermined threshold.

To understand the process better, consider the extension for the seed (MTA) from sequence1 and sequence2. For this example, we consider the ungapped alignment with match score = +1 and mismatch score = -1. Since it is a seed at the left corner of the sequence, it is extended to the right. The step by step extension of MTA is shown in Fig. 2.3. For a threshold of d = 4, the alignment of MTA is stopped at extension length = 15. The final alignment is obtained by trimming the alignment to a previously achieved maximum score, here 7. Fig. 2.4 shows how the alignment score varies with the alignment extension length, trimming and stop point.

The alignment till threshold and final alignment is given in Fig. 2.3.

Evaluation

In this step, the alignments obtained corresponding to the seeds are evaluated to determine if they are statistically significant based on the Karlin-Altschul

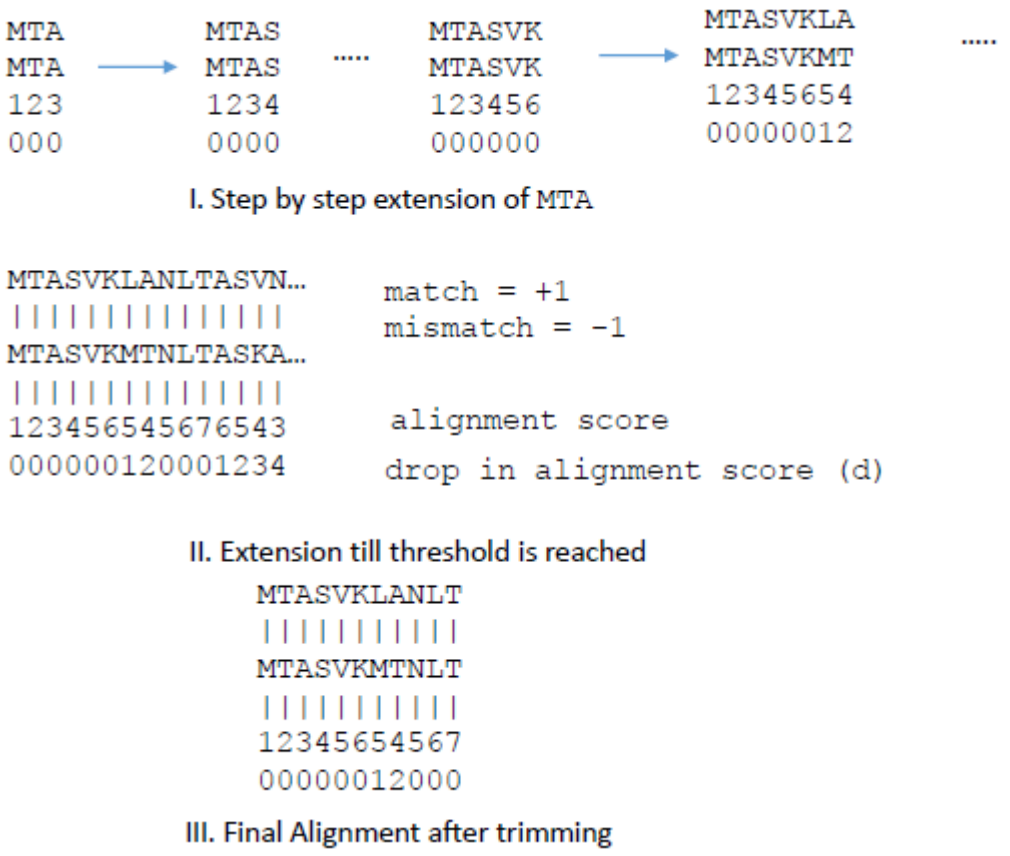


Figure 2.3: ***Extension: I. step by step extension of seed - MTA. II. Extension till predifined threshold is reached (here d = 4), III. Final alignment, trimmed at alignment score = 7.***

equation,

$$E = kmne^{-\lambda S}. \tag{2.1}$$

Here S is the raw alignment score; λS is the normalized score, where λ depends on the scoring matrices; mn is the search space size, where m is the number of letters in the query and n is the number of letters in the database (often called as database size) and k is the constant, usually set to 0.1. E gives the number of alignments (having normalized score = λS) that are expected by chance when a database of size n is searched for the sequence of length m. For example, if the alignment E-value is one, it means that one more hit with the same score is expected in the database purely by chance. The alignments that cross a prefixed threshold on E are considered statistically significant. Note that the speed and sensitivity of BLAST depends on various parameters including the word size, the neighbourhood size, search space

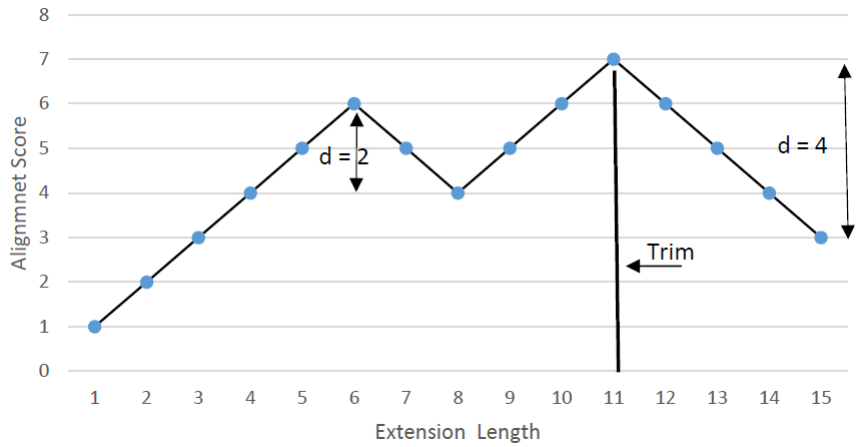


Figure 2.4: ***Extension length Vs Alignment Score. At d = 4, the extension is stopped, for final alignment, the alignment is trimmed at alignment score = 7.***

size, alignment-cut off threshold (d) and E-value. A comparison of the sensitivity of dynamic programming based method and fast heuristic methods like BLAST and FASTA is presented in [26] and [27]. Other post-BLAST heuristic techniques - BLAT [28] and UBLAST [29] are conceptually similar to BLAST and follow the "seed and grow approach", but outperform the original algorithm in speed of execution. The high execution speed of these methods is due to the careful use of an index structure of non-overlapping words in the database for seed selection, unlike BLAST which iterates through the database to find the seeds. Once the seeds for alignments is identified, in the follow-up step the seeds are extended to compute the alignments with a slight variation from the procedure used in BLAST. Although BLAT and UBLAST provide higher speed, they are less flexible when compared to BLAST as they only consider exact matches for seed selection. The flexibility offered by BLAST regarding seed selection makes it a popular choice as it provides good execution speed at the cost of tolerable loss of sensitivity.

Developing faster and accurate search algorithms remains an active area of research in-line with the ever growing availability of genomic sequences through advances in low-cost sequencing techniques. One of the latest such methods (published in year 2017) is MMseqs2. We use MMseqs2 and BLAST for comparing the search speed of our proposed method (refer to Chapter 5). The following section provide a brief description of the functioning of MMseqs2 [30].

2.1.2 MMseqs2

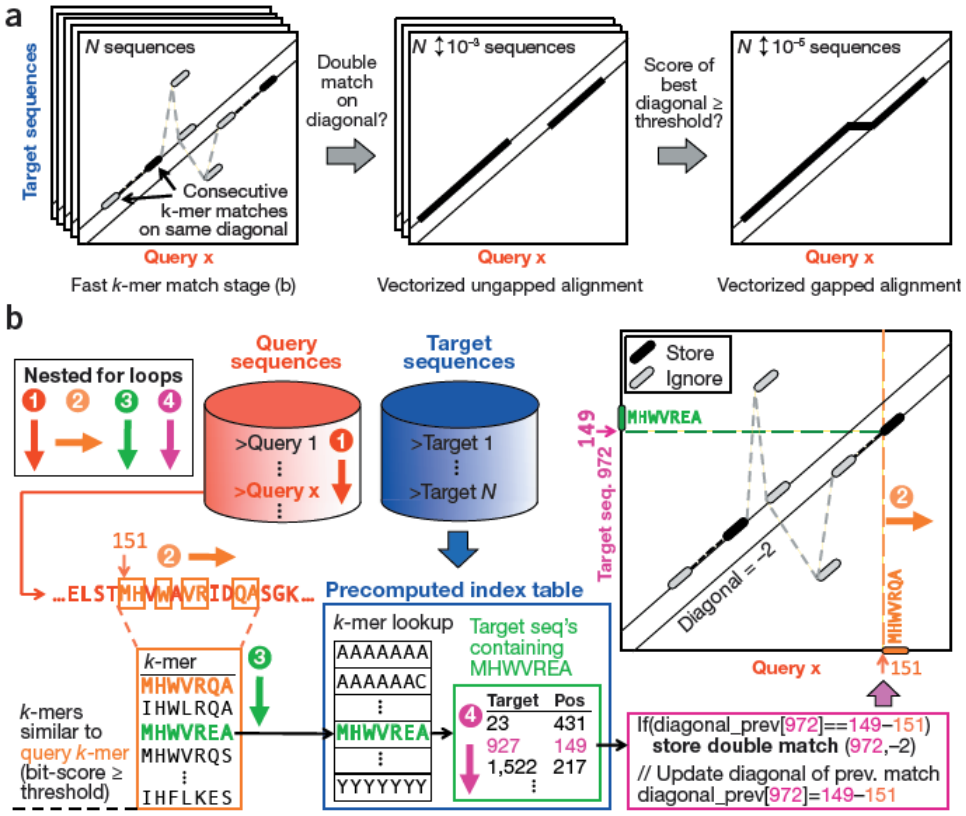


Figure 2.5: ***Description of MMseqs2 search algorithm. (This figure is reproduced from [30]) (a) Three search stages (i) fast k-mer search (ii) vectorized ungapped alignment (iii) gapped alignment; note that with each stage the search space/number of target sequences reduces. (b) Description of fast k-mer match stage. Loop1 iterate through queries in Query sequences. For query in loop1, here Query x, the corresponding sequence is split into k-mers of length 7 (loop2). For each k-mer in loop2, here MHWVRQA, the similar k-mers are determined. Each of these similar k-mers are looked up in a precomputed index table from the Database sequences (loop3). Here, the mapping of MHWVREA, containing the target ids and its position in the sequences is shown. In loop 4 the consecutive double matches are determined that are then used for producing the ungapped followed by gapped alignmnets.***

MMseqs2 algorithm is divided into three steps - (i) *k-mer* matching (ii) vectorized ungapped alignment and (iii) gapped (Smith-Waterman) alignment. In the *k-mer* match stage, for a given query sequence, the algorithm identifies target *sequences* that shares two consecutive *k-mers* on the same diagonal. Similar to BLAST, MMseqs2 relies on similar word matches. It uses a word size $k = 7$ and thus generates a large number of similar *k-mers*. Once consecutive *k-mers*

are identified in the diagonals, the ungapped alignment is computed. These ungapped alignments are further processed to compute the gapped alignments. The complete process adopted in MMseqs2 is summarized in Fig. 2.5. The speed achieved by MMseqs2 owes a lot to the vectorization provided for gapped and ungapped alignments; distribution of queries to multiple cores, and its ability to distribute the database to multiple servers. The software implementation details of MMseqs2 are provided in [30].

To date, the heuristics and native alignment-based methods dominate sequence analysis research, but these methods exhibit some limitations as listed below.

- Alignment based methods may not detect functionally similar sequences as they become more divergent.
- Alignment based algorithms are not robust under significant sequence rearrangements and they may produce an alignment score which does not reflect underlying biological similarity.
- Moreover, owing to the dynamic programming, the computational complexity involved in alignment based sequence comparisons is quadratic; for two sequences of length m and n the computational complexity is $O(mn)$ notwithstanding various acceleration strategies.

The limitations of alignment-based approaches and the explosion of biological sequence data arising from Next Generation Sequencing [31] has driven work toward approaches which do not rely on sequence alignment for similarity computation [32]. These efforts are collectively known as alignment-free approaches.

2.2 Alignment-Free Algorithms

Collectively, alignment-free algorithms can be summarized as "algorithms which do not rely on the computation of sequence alignment and hence the alignment score for evaluating the similarity of two sequences". Since these methods do not involve alignment and hence do not require dynamic programming, their time complexity may be significantly better than for alignment-based approaches. Development of alignment-free algorithms goes back to at

least 30 years when Blaisdell et al. [33] introduced an alignment-free measure for similarity computation. A lot of research since then focused on developing alignment-free methods for bioinformatics problems. The "Alignment-free" term is generic and is used for any algorithms/methods which do not involve sequence alignment for similarity computation. Reviews by Vinga et al. [2, 34] categorize alignment-free methods in two categories: (i) methods based on the frequency of subsequence (word-based methods) and (ii) methods based on the information content of the sequences (information-theoretic approach). Many other available methods which can't be classified into these two groups include methods based on chaos game representation [35], iterated maps [36], graphical representations [37, 38], length of matching words [39, 40, 41, 42]. In subsections below, we briefly discuss the word and information theory based alignment-free methods.

2.2.1 Word frequency-based methods

These methods are based on two assumptions: (i) similar sequences should share similar *sub-sequences* (also known as *k-mers*/words) (ii) the extent of shared *sub-sequences* should reflect the degree of similarity. The common steps involved in these methods are listed below:

- *Sub-sequence* extraction

  In this step, the sequence is sliced into *sub-sequences* by traversing a window (generally an overlapping window) of length k across it. These *k-mers* can also be thought as words of a text document. The analogy of sequences as documents and the *k-mers* as words motivates the use of text processing methodologies for bioinformatics problems.

- Mapping to a Vector embedding space

  In this step, the list of *k-mers* generated from a sequence is mapped to a vector space. Having fixed-length vector representations of sequences makes it possible to utilize machine learning (ML) algorithms for bioinformatics problems. Note that machine learning algorithms operate in a vector space; the performance of ML algorithms is affected by the quality of input used. Although the quality of a vector representation is particular to the application for which they are employed, in general, the

quality of representations is directly related to the number of discriminative features captured in the vector. The other important motivation of using vector representation for sequences is the availability of formal distance metrics like Euclidean or Hamming distance in vector spaces. These formal distance metric can be readily utilized for similarity calculation between sequences.

- Quantification of (dis)similarity

  In this last step, the formal metrics are used to compute (dis)similarity of sequences in the vector embedding space. The distance between two sequences in the vector space is considered as an equivalent to their alignment score.

To elaborate these steps, a toy example is as follows: Let Seq1 = CTACGCGC and Seq2 = AGCGCTAC be two DNA sequences. Slicing Seq1 and Seq2 using overlapping window of size, $k = 3$, we generate corresponding list of *k-mers*: (i) kmerList1 = [CTA, TAC, ACG, CGC, GCG, CGC] and (ii) kmerList2 = [AGC, GCG, CGC, GCT, CTA, TAC]. There can be several ways to map these lists of *k-mers* in a vector space (second step). A simple way of mapping the sequences in vector space is by counting the occurrence of constituent *k-mers*. In this example, let us assume that the unique words (lexicographically arranged) from kmerList1 and kmerList2 form a dictionary i.e. Dict = [ACG, AGC, CGC, CTA, GCG, GCT, TAC]. Practically, for DNA sequences, for $k = 3$, dictionary will have 64 unique words. To map kmerList1 and kmerList2 into a vector space we can count the *k-mers* from Dict in these lists. In this example, we finally obtain vectors: V1 = (1,0,2,1,1,1,1) and V2 = (0,1,1,1,1,1,1) for Seq1 and Seq2 respectively. Lastly, a distance metric can be used to calculate the (dis)similarity between Seq1 and Seq2.

Many methods are developed in this area by modifying the steps mentioned above. In the first step, the choice of k plays an important role; it is established in the available literature that in practice k in the range of $2-6$ is a good choice for protein sequence comparison for different phylogenetic distances [43, 44]. For DNA sequence analysis, k in the range of $9-14$ [45, 46] can be safely used. In their papers, Wang et al. [46] and Wu et al. [47] noted that for problems where sequences are less similar, a small k and problems involving

divergent sequences, larger k gives better results. Apart from *k-mer* selection, some methods also employ grouping of alphabets based on their chemical and physical properties [46] and eventually work with a smaller dictionary. The second step, mapping of sequences to vector space is a flexible operation, and different ways ranging from weighting to normalizing and clustering [48] have been employed for generating sequence representation. Such mapping also makes it possible to employ metrics like Euclidean, Manhattan distance and Pearson Correlation coefficient [2] as a measure of similarity.

2.2.2 Information theory based methods

These methods work on the principle of conservation of information contained in the sequences, where the extent of the closeness of information content of the sequences is considered to reflect the degree of similarity of the pair. Overall, these methods employ two-step process: in the first step, information content of the is represented using an information theoretic quantitiy such as complexity or entropy. In the second step, information-theoretic distances are calculated to quantify the difference in complexity/entropy and hence the (dis)similarity of the sequences. For example, the Kolmogorov complexity may be used to compute the information contained in the sequences. Note that for practical use Kolmogorov complexity is approximated using a compression algorithm and the length of the compressed sequences is used as an estimate of the complexity [49]. Other than this, Lempel-Ziv complexity [50] is a popular measure to calculate the information content. Once the complexity is computed, the normalized compression distance [51] may be seen as the measure of (dis)similarity of sequences.

Entropy is also widely used to represent the information content of sequences. In recent years, these concepts have been employed successfully in bioinformatics research for computing the coverage, block entropies and in prediction of transcription factor binding sites [34]. From the discussion above, it can be easily observed that the use of alignment-free methods for solving problems in bioinformatics has increased in recent years. This trend may reflect some of the advantages we have considered earlier for example (i) less computational complexity, (ii) avoidance of evolutionary assumptions and (iii) robustness in the face of structural rearrangements. We should be cautious in discarding

alignment-based approaches altogether, as for many bioinformatics applications like annotation of conserved protein domains, reconstruction of ancestral DNA sequences, alignment-algorithms still remain the best approaches.

In this work, we focus only on word-frequency based alignment-free methods. One crucial aspect of these methods is the "mapping of sequences" to vector space. Although such mapping offers various advantages as noted above, but there are some limitations, these are considered below:

- Exponential rise in dimensionality

  The dimensionality of the word-frequency vector increases exponentially with the choice of k. For example, to represent a DNA sequence with $k = 10$, the vector length is already 4^{10}. This problem is more severe for protein sequences which have an alphabet size of 20; the choice of $k = 5$ for proteins will generate vectors of size 20^5. This dramatically increases the space and computational requirements of the algorithms.

- Limited context

  Another issue is that word-frequency representations do not capture contextual information (co-occurrence of *k-mers*) and are not robust to the rearrangement of *k-mers* of sequences. For example, two sequences producing the same list of *k-mers* in a completely different order will produce the same vector. This challenge is addressed to some extent by N-gram representations, where the frequency of a group of *k-mers* is considered instead of the singleton *k-mer*. N-gram representations may include the context from a sequence to some degree but at the expense of a substantially increased vector dimensionality.

  Considering again the example given above for a DNA sequence with $k = 10$, for $N = 2$ (i.e. a group of two *k-mers* are counted) the vector size become $4^{2\times10}$ as compared to 4^{10} (singleton case). For proteins this is even more of a problem.

The limitations of word-frequency representations of sequences motivate us to develop approaches which can produce low dimensional representations that are independent of k, alphabet size and can capture more contextual information. Inspired by the representation learning literature in text processing, in

this thesis we focus on developing alignment-free approaches which work on low dimensional sequence representations to solve bioinformatics problems. In the section below, we discuss the representation learning work, followed by a detailed description of two widely used representation learning (RepL) architectures - `Word2Vec` and `Doc2Vec`.

2.3 Representation Learning

The choice of data representation plays a crucial role in determining the performance of a machine learning algorithm. The reason for this lies in the fact that the representations can differ in their ability to hold a different set of descriptive features of variation in the data. Owing to this, in a machine learning framework, much of the effort goes into constructing a preprocessing pipeline to transform the data in a desirable form. The main limitation of such feature engineering is its inability to include all discriminative information in the representations. Also, feature engineering very much relies on human expertise and domain knowledge. These limitations have motivated the development of methodologies which lean more towards learning of representations, acquiring discriminative features of the data automatically rather than employing feature engineering for data representation.

The interest in framing problems as representation learning approaches has rapidly grown since the year 2010 and have been successfully used in various domains such as speech recognition and signal processing [52, 53, 54] , object recognition [55, 56, 57] and Natural Language Processing [58, 59, 60]. A detailed review of representation learning is provided in [61].

Advances in learning distributed representations for text processing have led to techniques that provide a semantically meaningful, dense vector representation of words present in a large corpus of documents. The best known of these approaches is `Word2Vec`, due to Mikolov et al. [18]. The fundamental principle of `Word2Vec` [62] lies in the distributional hypothesis [63]: co-occurring words also share a semantic relationship. `Word2Vec` captures co-occurrence information in word embeddings by employing a simple classification task wherein a word is predicted based on its context (the nearby words) and the representations are learned in a manner such that words with similar

meanings have a relatively smaller distance in the embedding space. Follow up works [64, 65, 66] extended the idea of word embedding to the learning of embeddings for the whole document, based on suitable combinations of word embeddings. The `Word2Vec` model is essentially a feed-forward neural network. In the section below, we first provide a brief description of the underlying architecture of feed-forward neural networks before discussing the `Word2Vec` architecture in detail.

2.3.1 Feed-Forward Neural Networks

Information in the Feed-Forward neural networks is passed from the input to one or more layers to the output without any feedback or recurrent connections and hence these networks do not have any memory. The naive architecture of feed-forward networks involves fully connected successive layers, which means that each node of a layer has a weighted connection with all nodes of the successive layer. Fig. 2.6 shows this basic architecture. As shown, it mainly has an input, hidden and output layer; the output of hidden and the output layer is passed through a predefined function known as the activation function. Generally, a nonlinear function such as the *sigmoid* ($f(x) = \frac{e^{-x}}{1+e^{-x}}$) or the *tanh* ($f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$) are used as an activation function.

The overall purpose of a neural network is to approximate some function g as a function g^* by learning its parameters using known data. For example, the classifier, $y = g(x)$ maps an input x to a category y. The network defines an approximate mapping $y = g^*(x; W)$ and learns the value of the parameters - here the weights ($w_{11}, w_{12} \cdot \cdot$), generally represented as weight matrix (W) - that result in the best function approximation [67]. Learning of W is generally referred as training the neural network (NN). During the training process, the weights are randomly initialized and later updated usually using gradient-descent approach [67]. Once the model is trained, it is employed to give output for any new input. For example, if a classifier is trained with the known images of cats and dogs, once trained, it can be employed to classify any unseen (new) image as cat or dog.

Feed-forward networks may be extended to include feedback or recurrent connections; such networks are called recurrent neural networks (RNN's).

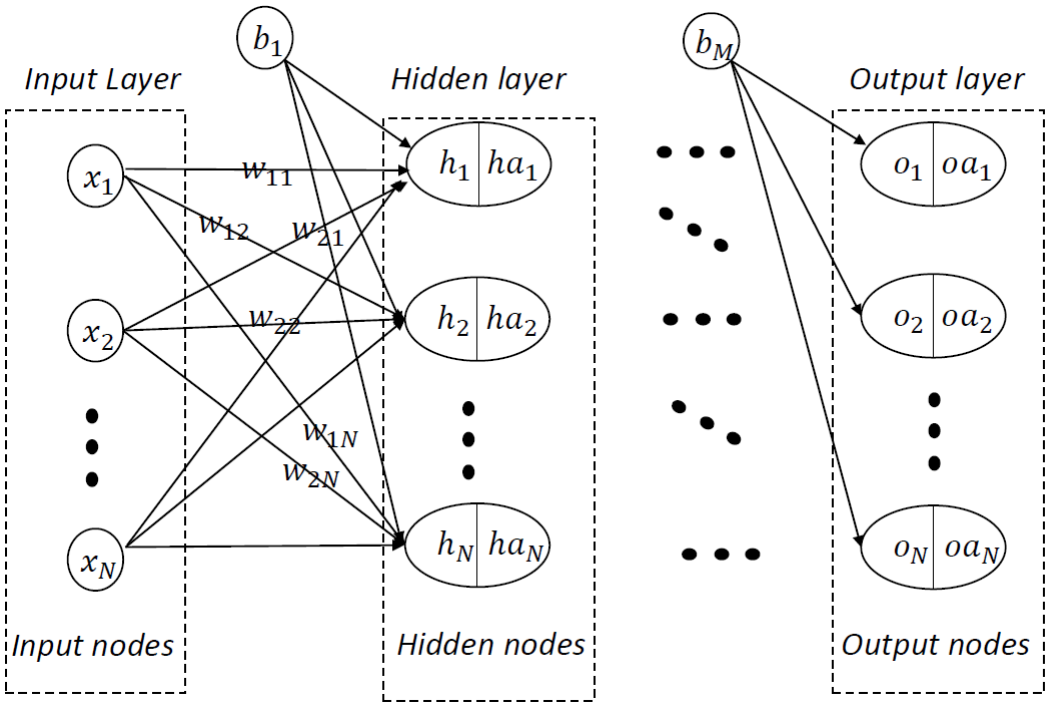


Figure 2.6: ***General architecture of a feed-forward neural network:*** x_i, h_i, o_i ***is the*** i^{th} ***input, hidden and output nodes;*** w_{ij} ***is the weight of a connection from*** i^{th} ***to*** j^{th} ***node;*** b_l ***is the bias of*** l^{th} ***layer, usually set to 1;*** ha_i, oa_i ***are the activated hidden and output node values obtained by passing the output at the hidden and output layer through activation function.***

Having feedback connections in the network equips the RNN with memory elements which make it useful for capturing the context presents in the sequential data such as text, sequences, etc. Unlike RNNs, `Word2Vec` and `Doc2Vec` provide a way to capture the contextual information in word and document embeddings using a feed-forward NN architecture as discussed below.

2.3.2 `Word2Vec` Architecture

The idea behind `Word2Vec` is that pairs of words which share a semantic relationship should be proximally located in the embedding space. `Word2Vec` achieves these embeddings through fully-connected shallow neural networks shown in Fig. 2.7. The input and the output layer of this network have $|V|$ nodes, where $|V|$ is the vocabulary size. Each node at the input/output layer is mapped to a vocabulary word such that the i^{th} node corresponds to the

i^{th} word in the vocabulary. The weight vector $\mathbf{v}_i \in \mathbb{R}^n$ denotes connections from the i^{th} input node to the hidden layer nodes, where n is the number of nodes in the hidden layer. The concatenation of the input node weight vectors forms the input-hidden layer weight matrix, $\mathbf{W} \in \mathbb{R}^{|V| \times n}$. Similarly, the hidden-output layer weight matrix is denoted $\mathbf{W}^{'} \in \mathbb{R}^{n \times |V|}$.

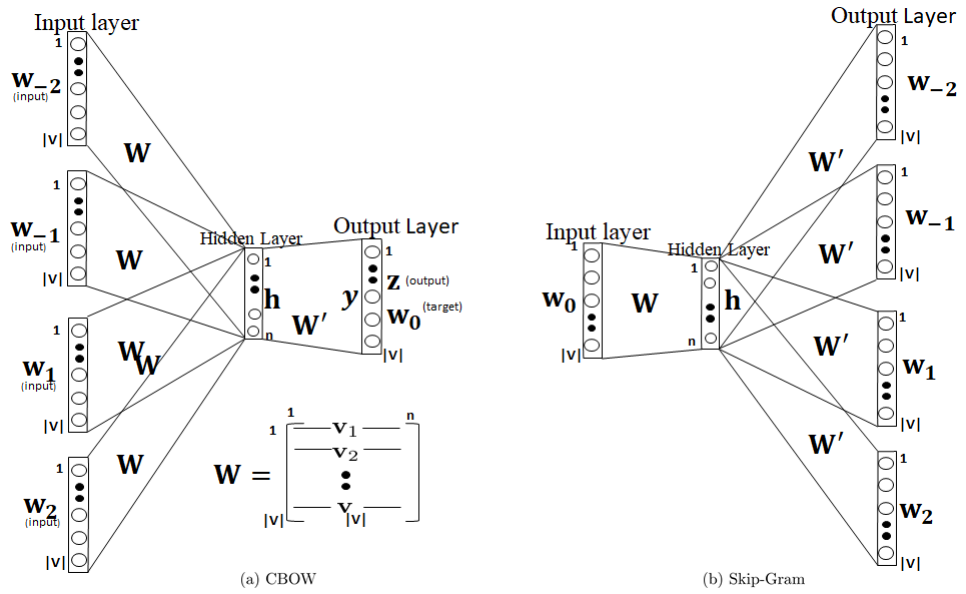


*Figure 2.7: `Word2Vec` **architectures: (a) CBOW - uses context (i.e. adjacent words) to predict central word, here w_0 and (b) Skip-Gram - uses central word to predict context.***

Once training is completed, the weight vector $\mathbf{v}_i$ corresponding to the i^{th} word at the input/output layer is also its corresponding embedding. These embeddings are learned such that words with semantically similar meanings are closer in the embedding space. This objective is achieved by a prediction based framework, where the loss function of the NN is a likelihood function for the words in the corpus. The likelihood function is proportional to the probability of the occurrence of a word given its context. `Word2Vec` method has two variants, namely the CBOW (Continuous Bag of Words; Fig. 2.7 (a)) and the Skip-gram Fig. 2.7 (b)). In the CBOW architecture, the context is given at the input to predict the word, whereas in the Skip-gram architecture, the word is used to predict the context.

Fig. 2.7 shows a sample $w_{-2}, w_{-1}, w_0, w_1, w_2$. Here, w_0 denotes the word at the center and w_i $(i \neq 0)$ denotes the nearby context words. The subscript i

denotes the position of words relative to the central word, w_0. Inputs to the NN are here a one-hot-encoding of the context words, as shown in small bold letters in the figure.

A context is represented in a binary vector by keeping all elements zero except the indices corresponding to its constituent words. Here we consider the CBOW architecture in detail. As shown in Fig. 2.7 (a), the one-hot-encoding of words in the context operates on the weight matrix $\mathbf{W}$ to give the hidden layer vector $\mathbf{h}$,

$$\mathbf{h}^T = \left[\mathbf{w}_{-2} + \mathbf{w}_{-1} + \mathbf{w}_1 + \mathbf{w}_2\right]^T \mathbf{W}. \tag{2.2}$$

Here, $\mathbf{w}_{-2} + \mathbf{w}_{-1} + \mathbf{w}_1 + \mathbf{w}_2$ is the vector corresponding to the context. Similarly, the input to the output layer is given as $\mathbf{y}^T = \mathbf{h}^T\mathbf{W}^{'}$. Finally the *Softmax* function is applied at the output layer to produce the output vector $\mathbf{z} = softmax(\mathbf{y})$. The output at node j is thus

$$z_j = \frac{\exp(\mathbf{y}_j)}{\sum_{k=1}^{V} \exp(\mathbf{y}_k)}. \tag{2.3}$$

Here, z_j, y_j are the j^{th} elements of the vectors $\mathbf{z}$ and $\mathbf{y}$ respectively; note that the target vector is $\mathbf{w}_0$. Vector $\mathbf{z}$ can be interpreted as the probability distribution of all words in the vocabulary given the context of w_0 at the input. The embeddings for words in the corpus are obtained by maximizing the joint probability of a word, w, conditioned on its context, C, for all samples in the corpus. The conditional probabilities for each sample are assumed to be independent of each other and hence the joint probability for all samples can be written as, $\prod_{w \in Vocab} \Pr\left[w \mid C\right]$. Maximizing the $\prod_{w \in Vocab} \Pr\left[w \mid C\right]$ is the same as minimizing its negative log likelihood, so the overall loss function for `Word2Vec` may be written as:

$$\mathrm{J}(\mathbf{W}) = \sum_{w \in Vocab} -\log \Pr\left[w \mid C\right]. \tag{2.4}$$

We note that updating the weight vectors implies adjusting those vectors corresponding to words in the embedding space. Maximizing the probability $\Pr\left[w_0 \mid C\right]$, as given in Eq. 2.3, is equivalent to maximizing the dot product of the weight vectors of w_0 and its context words, ultimately reducing the cosine

distance between them. The process of maximizing the conditional probabilities (Eq. 2.4) therefore translates to reducing the cosine distance in the embedding space between the word and its surrounding context. The resulting optimisation problem can be solved using the standard gradient descent approach. However, to increase the computational efficiency of the algorithm, Mikolov et al. in [62] use a negative sampling method that essentially modifies the original objective function so that each training sample only updates a few weight vectors.

2.3.3 `Doc2Vec` Architecture

The `Doc2Vec` model essentially inherits the advantages offered by the `Word2Vec` model and generates a low dimensional representation of each document and its constituent words. These representations can then be used to retrieve documents from a collection. The main difference in the architectures of `Doc2Vec` and `Word2Vec` lies in use of a document tag[2] along with the underlying words. The `Doc2Vec` architecture thus requires additional nodes to specify the document tag as input. However, the output layer remains unchanged, and after training is completed, the `Doc2Vec` model yields both word embeddings and embeddings for the documents.

2.3.4 Representation Learning approaches in bioinformatics

Advances in word embedding methods in text processing have inspired several studies of sequence encoding in the bioinformatics domain. The earliest work was due to Asgari et al. [20], where they introduced `BioVec`, an adaptation of the `Word2Vec` framework to provide embeddings for molecular sequences. They used the skip-gram variant of `Word2Vec` to obtain *k-mer* embeddings; to provide sequence embedding, they propose to sum the embeddings of all the *k-mers* present in the sequence. However, `BioVec` does not preserve the *k-mer* ordering of the sequence. In follow up work, we addressed this limitation through `Seq2Vec` [68], which relies on the `Doc2Vec` [64] architecture to provide a direct embedding for the sequence. This work is considered in detail in chapter 3.

[2] A label attached to document for its identification.

While the application of Word2Vec and Doc2Vec to biological sequences is rather straightforward, the insights obtained are not. Both of these studies confirmed the utility of embeddings in capturing relevant features of protein sequences — initially using BioVec and through subsequent improvements in Seq2Vec. The BioVec study also provided a crucial insight regarding the relationship among *k-mer* embeddings and the biochemical and biophysical properties of the sequence: mass, volume, and charge. It was shown that *k-mers* with similar physico-chemical properties form clusters in the embedding space. Further, in the Seq2Vec paper we demonstrated that the use of a sequence tag during learning results in even better clustering of the sequence data – sequences belonging to the same family were more likely to be clustered together using Seq2Vec than with the BioVec model. Such representations have proven beneficial for classification and retrieval tasks.

Following the BioVec and Seq2Vec approaches many other works were proposed to address different bioinformatics applications, such as predicting gene-gene interactions [69], splice junction prediction [70] and protein function prediction [71]. Some of these works include DNA2Vec [72], Gene2Vec [69], SpliceVec [70] and Dom2Vec [71]. In DNA2Vec [72], the authors proposed a method for learning the distributed representation of variable length *k-mers*. They demonstrate that adding DNA2Vec nucleotide vectors is the same as nucleotide concatenation. Further, they also establish a correlation between the Needleman-Wunsch similarity score and cosine similarity of the DNA2Vec generated vector representations. In Gene2Vec [69], the authors provide a method for learning distributed representations of genes based on their co-expression. They demonstrate the application of learned gene representations for predicting gene-gene interaction. In SpliceVec [70], the authors utilized the word embedding approach for learning the representation of the splice site and further used these representations for the splice junction prediction task. The most recent work, Dom2Vec [71], tries to explore the embeddings of protein domains for protein classification tasks. Representation learning in bioinformatics is still in its infancy and there has been an ongoing attempt to provide better generic/task-specific embeddings for sequences/amino acids. In a recent work by Bepler et al. [73], the authors proposed a method to improve the sequence embeddings by incorporating structural information.

These embeddings were shown to provide the best result for structural similarity prediction and transmembrane prediction as compared to alignment-based approaches. Other recent papers ([74], [75] and [76]) explore language modelling approaches (particularly based on the RNN and its variants) for providing sequence and amino acid representations, but these approaches lie outside the scope of the present work.

2.3.5 Conclusion

This chapter sets the context for the works pursued in this thesis - covering the relevant literature from sequence analysis and representation learning research. Here we provided a review of alignment-based and alignment-free methods and noted their advantages and limitations. We specifically covered details of two well-known alignment-based approaches - BLAST and MMseqs2, that we used to benchmark our work on homologous sequence retrieval task (refer to chapter 5). In the later sections, we reviewed representation learning research, including the details of `Word2Vec` model - which inspired architectures of the representation learning models proposed in this thesis. Lastly, we covered recent publications concerning representation learnings research from bioinfromatics domain.

Having set the background and context of the research work pursued in this thesis, we provide the details of our proposed approaches and their application to the particular problems in upcoming chapters. Next chapter covers our first work on learning distributed representations for biological sequences.

3 Seq2Vec: Distributed Representation Learning for Biological Sequences

Overview

Representing biological sequences as mathematical entities (vectors or matrices) is a fundamental preprocessing step for many machine learning (ML) based solutions for bioinformatics problems. The performance of these solutions is affected by the representations which are given as input to machine learning algorithms. Improving these representations has always been of interest to the researchers working in the bioinformatics domain to enhance the performance of ML-based methods on bioinformatics tasks. The most prevalent representations used for biological sequences are based on the bag-of-words technique from text processing. The main limitations of these representations are (i) their inability to capture contextual information and (ii) their high dimensionality. To address these limitations, we propose an alternative data-driven approach that learns the patterns hidden in the sequences and generates low dimensional distributed representations for them. We call this framework, `Seq2Vec` - its architecture is inspired by document embedding approach `Doc2Vec` from text processing domain (refer section 2.3.3).

In the previous chapter, we covered the relevant topics from sequence analysis and representation learning research. We specifically provided details of two sequence comparison tools, BLAST and MMseqs2. Also, we looked at most popularised representation learning approaches from text processing domain - `Word2Vec` and `Doc2Vec`.

In this chapter, we provide details of `Seq2Vec` and demonstrate its utility for the protein family classification task. We compare `Seq2Vec` performance against an earlier representation learning approach - `BioVec` - and also against the standard alignment-based approach - BLAST. The organisation of this chapter is given below.

We begin in section 3.1.1 by outlining the notation to be used in this and subsequent chapters. Section 3.1.2 is devoted to the pre-processing step in which sequences are converted into an array of sub-sequences an important prerequisite for the discussion of the architecture and function of `Seq2Vec` (section 3.2).

Section 3.3 describes the experimental setup that will be used to evaluate the performance of the method. This includes a brief discussion of protein families (section 3.3.1), description of the data used for experiments (section 3.3.2), and further discussion of parameter selection (section 3.3.3) and evaluation metrics (section 3.3.4). Results of these experiments are provided in section 3.4 and this includes the performance for both binary and multiclass classification problems.

An important aspect of representation learning methods is to consider the quality of the embeddings which are generated. In section 3.4.3, we discuss the quality of the embeddings generated by `BioVec` and `Seq2Vec` and evaluate them in terms of the inter-class and intra-class distances in the embedding space. This is done by analysing neighbourhood-based classification results. In section 3.4.4 we compare the classification results obtained using an embedding based approach with an standard method based on search of families based on BLAST.

Finally, we conclude the chapter in section 3.5.

3.1 Preliminaries

3.1.1 Notation

We denote the corpus of N protein sequences by $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$ and the vocabulary with M unique *k-mers* as $\mathcal{K} = \{k_1, k_2, \ldots, k_M\}$. Each sequence

$s_i = \left[k_{i_1}, k_{i_2}, \ldots, k_{i_{n_i}}\right]$ is an ordered list of *k-mers*. To avoid notational clutter we use s_i to also denote its tag[1]. Each sequence in $\mathcal{S}$ belongs to one of the L classes (here protein families) $l_1, l_2, \ldots, l_L$ and $s_i \in l_k$ means l_k is the label for sequence s_i. The vector representations of sequences and *k-mers* are shown in **bold letters**, for example, vector representation (column vector) for s_i and k_i is given as $\mathbf{s}_i$ and $\mathbf{k}_i$ respectively. Sequence and *k-mer* embedding matrices $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N]^T$ and $\mathbf{K} = [\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M]^T$ denote matrices containing all the sequences and *k-mer* vectors as rows. Finally, we write the sigmoid function as:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}. \tag{3.1}$$

3.1.2 Preprocessing

Our proposed approach `Seq2Vec` is inspired by the `Doc2Vec` framework proposed in the text-processing domain. `Doc2Vec` generates representations for documents capturing contextual information and, at least to some extent, the ordering of words. To apply similar concepts for learning representations of sequences, we first preprocess sequences to obtain a document like structure. Unlike a document, which may be seen as an array of words with a certain linguistic structure, biological sequences are strings of letters selected from an alphabet – for DNA sequences, {A, C, G, T}, the set of nucleotide Adenine, Cytosine, Guanine and Thymine, while for protein sequences the alphabet consists of 20 letters, each representing an amino acid. Since there is no clear notion of words in biological sequences, we need to break these sequences during the preprocessing stage to represent them as an array of *k-mers* – a unit made of k consecutive letters.

K-mers are extracted by sliding a window of size k across the sequence from left to right. We may generate either overlapping or non-overlapping *k-mers*. In the case of non-overlapping *k-mers*, we generate k new sets of *k-mers* from the given sequence by shifting the starting point at the beginning of the sequence. For example, given a hypothetical sequence,

QWERTYQWERTY

[1] A special symbol attached to the sequence for its identification.

and $k = 3$, we generate three new sequence sets as follows:

Seq 1: QWE RTY QWE RTY
Seq 2: WER TYQ WER
Seq 3: ERT YQW ERT.

In the case of overlapping *k-mers*, we generate a single set by breaking the original sequence into overlapping *k-mers*. While the overlap itself is a parameter, in this work we focus as usual on *k-mers* shifted by one letter at a time. For instance, the hypothetical sequence above yields following state of overlapping *k-mers*:

QWE WER ERT RTY TYQ YQW QWE WER ERT RTY.

Having discussed the notation and preprocessing step to generate a preliminary bag-of-words representations, we now discuss in detail the architecture and function of the proposed approach, `Seq2Vec`.

3.2 Proposed Approach: `Seq2Vec`

In one of the first attempt to use representation learning approaches from text processing for sequence analysis, Asgari et al. [20] applied the `Word2Vec` framework to learn the embeddings for *k-mers* calling the approach `BioVec`. They successfully established that *k-mers* that share similar physico-chemical properties form clusters in the vector space. Further, to generate vector representations for protein sequences, they proposed to add the embeddings of the constituent *k-mers*. One of the potential weaknesses of such an approach is that it does not *fully* capture the order in which the *k-mers* occur in the original sequence and hence may give the same representation for two sequences with different structures. For example, two sequences that have same *k-mers* in different order will have same `BioVec` representation. We address this issue through our proposed methods - `Seq2Vec` which is based on the `Doc2Vec` approach. Unlike `BioVec`, `Seq2Vec` estimates the representations of sequences considering both *order* and contextual information.

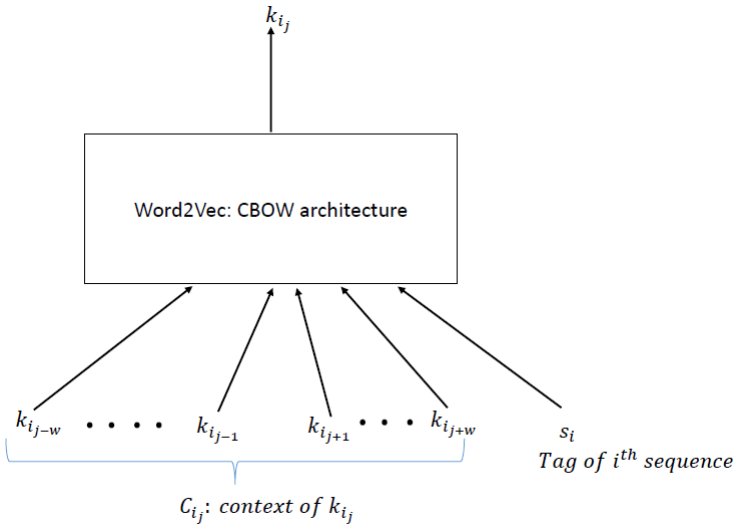


Figure 3.1: `Seq2Vec` **Model*: It is a shallow neural network with the CBOW variant of `Word2Vec` model. The context of k_{i_j} comprises its adjacent words and is denoted $k_{i_{j-w}}$, ..., $k_{i_{j-1}}$, $k_{i_{j+1}}$....$k_{i_{j+w}}$.***

`Seq2Vec` is essentially an embedding technique based on a shallow neural network that takes as input a set of *k-mers* extracted from a set of biological sequences. The method produces a representation in an Euclidean space of a chosen dimension. The core idea of `Seq2Vec` is to capture the contextual information available and to learn an abstract representation of each sequence. This is achieved by training the model via a simple classification task, in which each *k-mer* is predicted from its context and the tag of the corresponding sequence. The representations are learned by iterating through the set of sequences. For example, in Fig. 3.1, the prediction is shown for a sample from the i^{th} sequence in the corpus. Here the sample consists of the j^{th} *k-mer* k_{i_j}, the context C_{i_j} and the tag of the corresponding sequence, s_i. Here C_{i_j} is given as:

$$C_{i_j} = \{k_{i_{j-w}}, \ldots, k_{i_{j-1}}, k_{i_{j+1}} \ldots, k_{i_{j+w}}\}. \tag{3.2}$$

We combine the vectors of all the *k-mers* of a particular context by adding them together as:

$$\mathbf{C}_{i_j} = \left(\sum_{r=-w}^{w} \mathbf{k}_{i_{j+r}} \right) - \mathbf{k}_{i_j}. \tag{3.3}$$

Here $\mathbf{C}_{i_j}$ is the vector corresponding to C_{i_j}. `Seq2Vec` model is trained by updating its parameters so as to maximize the probability of a *k-mer* given its context and corresponding sequence tag for all samples in the corpus. For the sample constituting k_{i_j}, C_{i_j} and s_i the conditional probability is written as:

$$\Pr\left[k_{i_j} \mid C_{i_j},\ s_i\right], \tag{3.4}$$

which is defined using the softmax function as:

$$\Pr\left[k_{i_j} \mid C_{i_j},\ s_i\right] = \frac{\exp\left[\langle \mathbf{k}_{i_j},\ \mathbf{h}_{i_j}\rangle\right]}{\sum_{k\in\mathcal{K}} \exp\left[\langle \mathbf{k}, \mathbf{h}_{i_j}\rangle\right]}. \tag{3.5}$$

Here $\mathbf{h}_{i_j}$ is combination of context vector, $\mathbf{C}_{i_j}$ (defined in Eq. 3.3) and the sequence vector $\mathbf{s}_i$, as defined below:

$$\mathbf{h}_{i_j} = \frac{\left(\sum_{r=-w}^{w} \mathbf{k}_{i_{j+r}}\right) - \mathbf{k}_{i_j} + \mathbf{s}_i}{2w+1}, \tag{3.6}$$

where w is the context size.

Maximizing the conditional probability defined in Eq. 3.4 is the same as minimizing its negative logarithm. When considered for all the samples in the corpus of N sequences, the overall negative log-likelihood function can be written as:

$$\mathrm{J}(\mathbf{S},\mathbf{K}) = \min \sum_{s_i\in\mathcal{S}} \sum_{j=1}^{n_i} -\log\ \Pr\left[k_{i_j} \mid C_{i_j},\ s_i\right]. \tag{3.7}$$

Note that to minimize $\mathrm{J}(\mathbf{S},\mathbf{K})$ we need to compute the softmax probability (Eq. 3.5) for each *k-mer* in the vocabulary, which makes it practically infeasible. Taking hints from the literature on `Word2Vec`, we use the hierarchical softmax [77] (HS) approach to evaluate the expression and hence reduce its computational complexity from $\mathcal{O}(\mathcal{K})$ to $\mathcal{O}(\log_2 \mathcal{K})$. HS is based on a binary tree ($\mathcal{T}$) construction, where each leaf maps to a *k-mer* in $\mathcal{K}$. Also, since it is a

binary tree, any leaf (*k-mer*) can be reached rapidly from the root by traversing a unique path. We denote the q^{th} node in such a path from the root to the leaf *k-mer* k as $n(k, q)$ and the corresponding vector as $\mathbf{n}(k, q)$. An example of the binary tree highlighting the path from the root to *k-mer* k_2 and the intermediate nodes is shown in Fig. 3.2.

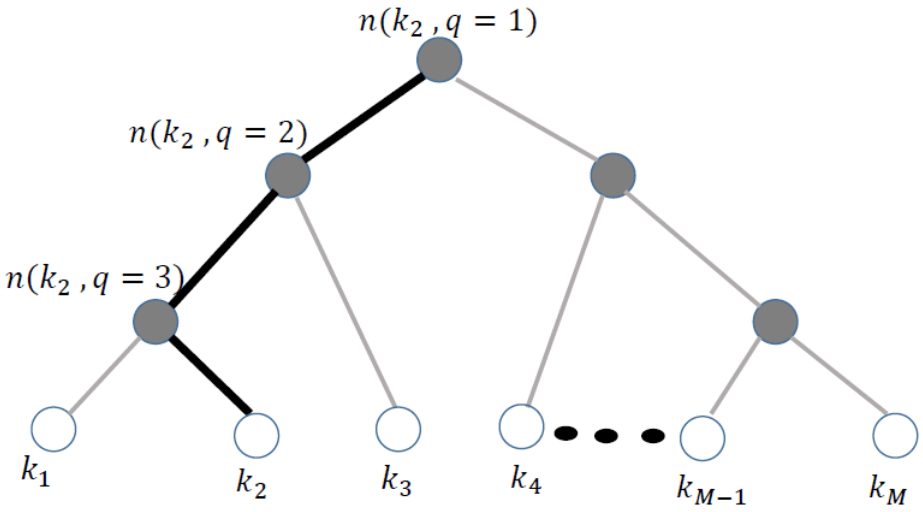


Figure 3.2: ***An example to demonstrate binary tree that is used to compute conditional probabilities using hierarchical softmax. Each of the leaf node corresponds to* k-mer *in the vocabulary that can be traced following a unique path from the root.***

Considering a stack of these internal node vectors of $\mathcal{T}$ yields the internal node matrix, $\mathbf{G}$. The probability for a specific *k-mer* k_{i_j} is obtained by traversing from the root to the leaf corresponding to k_{i_j} and multiplying the contributions (defined below) from each internal node in the path. Also, the nodes are weighted based on their location, corresponding to the parent node. If the node is the left child, it is weighted as 1; otherwise -1. For the example shown in Fig. 3.2, $\Pr\left[k_2 \mid C_{i_j},\ s_i\right]$ is calculated as:

$$\sigma\left(\langle \mathbf{n}(k_2, q=1), \mathbf{h}_{i_j} \rangle\right) \cdot \sigma\left(\langle \mathbf{n}(k_2, q=2), \mathbf{h}_{i_j} \rangle\right) \cdot \sigma\left(\langle \mathbf{n}(k_2, q=3), \mathbf{h}_{i_j} \rangle\right), \quad (3.8)$$

here $\langle \cdot \rangle$ denotes dot product. Generalizing it for sample k_{i_j}, C_{i_j} and s_i, the conditional probability in Eq. 3.5 using HS is calculated as

$$\prod_{q=1}^{L_{i_j}-1} \sigma\left(\text{child}(q-1) \cdot \langle \mathbf{n}(k_{i_j}, q), \mathbf{h}_{i_j} \rangle\right), \tag{3.9}$$

where $\text{child}(q-1) = 1$ if $n(k_{i_j}, q)$ is the left child of $n(k_{i_j}, q-1)$ and -1 otherwise. Here L_{i_j} is the length of the path i.e. the total number of nodes in the path from the root to the j^{th} *k-mer* of sequence s_i.

Finally, we can expand the function in Eq. 3.10 as

$$\text{J}(\mathbf{S}, \mathbf{K}, \mathbf{G}) = \min \sum_{s_i \in \mathcal{S}} \sum_{j=1}^{n_i} -\log \prod_{q=1}^{L_{i_j}-1} \sigma\left(\text{child}(q-1) \cdot \langle \mathbf{n}(k_{i_j}, q), \mathbf{h}_{i_j} \rangle\right) \tag{3.10}$$

which is minimized following the gradient-descent approach to train the `Seq2Vec` model.

Once trained, the updated vectors in $\mathbf{S}$ correspond to the representations of sequences. For a new sequence, the representation is obtained using gradient descent on the objective function (Eq. 3.10), while fixing everything else but the sequence vector to be learned.

The embeddings for sequences obtained through `Seq2Vec` can further be used for downstream bioinformatics tasks. In this chapter, we show its application for the protein family classification task, discussed in the sections below.

3.3 Experimental Setup

To demonstrate the advantages of `Seq2Vec` we focused on family assignment problem. Since the coding region of DNA sequences contains some redundancy, we worked directly with protein sequences, as we expected that this would reduce the noise level and improve the quality of the embeddings. Nucleotide gene sequences can be assigned to their appropriate family using this approach to translate the DNA to the corresponding protein sequence.

For experiments we compare it with an earlier representation learning method `BioVec` [20] which was briefly introduced at the beginning of section 3.2. Unlike `Seq2Vec`, `BioVec` computes the representation for individual *k-mers* which are further combined to obtain the embeddings for the overall sequence.

We carefully designed our experiments to compare the two sets of embeddings on multiple fronts. First, we use the protein vectors learned via `Seq2Vec` and `BioVec` [20] as features and compare them for the task of protein classification using SVMs. We consider both binary and multiclass classification problems. Next, since distributed representations embed similar sequences in proximity to each other, we use the k-nearest neighbours (kNN) to retrieve the k closest sequences (based on the euclidean distance) in the vector space and to see how successful are we in predicting the family of a test sequence based on a majority vote. Finally, we compare this retrieval task with BLAST, the standard approach for rapid determination of sequence similarity, and notice that the performance gap between BLAST and embedding based retrieval widens with increasing k (neighbourhood size).

Before presenting the results in detail, in the sections below, we first briefly discuss grouping of proteins as families and the data used for experiments.

3.3.1 Protein Families

A protein family is a group of related proteins exhibiting significant structural similarity at both the sequence and molecular level. Large scale assignment of proteins to families grew out of the work of Dayhoff [78]. Sequence homology is regarded as a direct reflection of evolutionary relatedness, and members of the same protein family may also exhibit a similar secondary structure through common functional units called *domains*. Domains present an additional organizing principle, allowing protein families to be grouped into super families [79] based on their domains, even if sequence homology is limited for some pairs of protein entries. These families and super-families are documented in detail in databases such as SCOP [80] and their successors. Direct determination of 3D protein structure is difficult and time-consuming, requiring complex experimental techniques such as X-ray crystallography or NMR spectroscopy. For this reason, experimental confirmation is available for

only a small fraction of the proteins available, leading researchers to develop methods [81, 82, 83] which use only the primary structure for family and thus functional identification.

3.3.2 Data

For the experiments, we use the protein family data containing 324018 protein sequences from [20] which are taken originally from the Swiss-Prot database [84]; they also provided a metafile which includes the family labels along with other description of the protein sequences.

We extracted the family label information from the given metafile. To our surprise, we found 6967 unique family labels extracted from the metafile which differs than the total number of families (7027) reported by [20]. On further probing, we found that in the classification result file provided by [20] there are multiple entries for some families, but the number of samples corresponding to such entries is different. Adding up the number of sequences for such entries for a given family matches to the actual number of sequences as per the metafile. This suggests that the samples of a few families are split into sub groups by [20] in their experimental setup, which is the reason for difference in number of families reported in this work and [20]. In our experiments we utilized the labels from metafile and based on these labels categorized sequences into 6967 families out of these, 3861 families have fewer than 10 sequences, 2472 families have sequences in the range 11 - 100, 609 families have sequences in the range of 101 - 1000, and remaining 25 families contain more than 1000 sequences. Distribution of class (family) sizes and length of sequences is provided in Appendix A

Since multiple parameters need to be fixed for using `Seq2Vec` model, to choose the best parameters we performed classification experiments for different sets of parameters as discussed below.

3.3.3 Parameter selection

To choose a good set of hyper-parameters – including context size, vector space dimension, non-overlapping vs overlapping pre-processing, model architecture – we did a search over a range of these parameters for the task of

kNN based classification and chose parameters which gave the best accuracy on a separate validation set for sequence retrieval from the same family as the query sequence. We did this experiment using sequences from the 25 biggest families having more than 1000 sequences. Overall, from this experiment we chose dimension: 250, *k-mer* size: 3, context size: 5, pre-processing: non-overlapping (see Fig. 3.3). All these experiments were performed using Gensim [85]

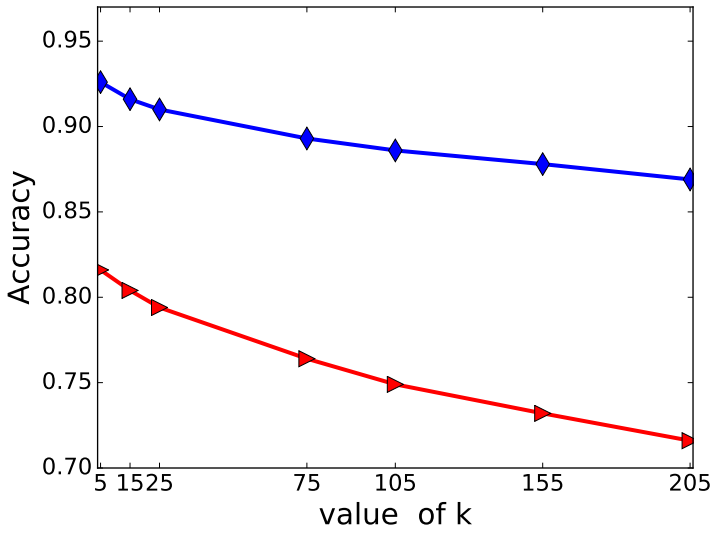


Figure 3.3: ***Performance of non-overlapping (–♦–♦–) vs overlapping (–►–►–) pre-processing for the task of kNN based protein family classification for different number of nearest neighbors. This is with best choice of other parameters – dimension:*** 250, **k-mer** ***size:*** 3, ***context size:*** 5 (25 ***for overlapping). It is clear that non-overlapping processing performs significantly better than the overlapping processing.***

3.3.4 Evaluation metrics

The metrics used to compare the performance of `Seq2Vec` and `BioVec` for the task of binary and multiclass classification are as follows:

$$\begin{aligned} \text{Specificity} &= \frac{\text{TN}}{\text{TN}+\text{FP}} \\ \text{Sensitivity} &= \frac{\text{TP}}{\text{TP}+\text{FN}} \\ \text{Accuracy} &= \frac{\text{TN}+\text{TP}}{\text{TN}+\text{TP}+\text{FN}+\text{FP}} \end{aligned}$$

where TN is the number of true-negatives, TP is true-positives, FP is false-positives, and FN is false-negatives. Specificity (true negative rate) measures the proportion of negatives that are correctly classified. Sensitivity (true positive rate or recall) measures the proportion of positives that are correctly classified. Accuracy measures the proportion of examples that are correctly classified. Depending on the application, the relative importance of these metrics may change.

Note that we use the vectors provided by authors of [20] for doing experiments involving `BioVec`[2].

3.4 Protein Family Classification Results

3.4.1 Binary Classification

This experiment exactly follows the setting used by [20]. The task is to distinguish sequences of a given family from all other families. For each family, the corresponding sequences constitute the positive class, and the same number of sequences, randomly chosen from the rest of the families, forms the negative class. A binary classifier is trained, and the metrics are computed using 10-fold cross-validation. Note that, since we have many classes with a small number of samples, making an independent test set is challenging. Hence, we adopted a cross-validation approach for evaluating the proposed method.

While the results reported in [20] also trained classifiers for families with fewer than 10 sequences, we included only families with at least 10 sequences in order to perform a proper 10-fold validation. Also, more data makes the learned classifiers more stable, and the reported results are stable and reproducible. The exact choice of SVM hyper-parameters, including the type of kernel and regularization parameter: C, is not mentioned in [20]. We chose to use an SVM classifier with linear kernel and C equal to 1.0, these choices being made based on grid search. We did three sets of experiments with the 1000, 2000, and 3000 biggest families. The results reported in Table 3.1 (values+s.dev) show that the accuracy and specificity obtained using

[2] http://llp.berkeley.edu/

	# of classes	Specificity (%)	Sensitivity (%)	Accuracy (%)
Seq2Vec	1000	**97.49 ± 0.05**	92.72 ± 0.06	**95.10 ± 0.05**
	2000	**97.01 ± 0.04**	93.07 ± 0.07	**95.04 ± 0.05**
	3000	**96.67 ± 0.05**	93.18 ± 0.08	**94.92 ± 0.06**
BioVec	1000	91.61 ± 0.05	**95.40 ± 0.04**	93.50 ± 0.05
	2000	90.10 ± 0.07	**95.75 ± 0.05**	92.92 ± 0.05
	3000	88.61 ± 0.09	**95.95 ± 0.05**	92.27 ± 0.06

Table 3.1: ***Binary Classification: performance of*** `Seq2Vec` ***and*** `BioVecs` ***for top 1000, 2000 and 3000 protein families. For each evaluation metric values and corresponding std.dev is noted.***

`Seq2Vec` is consistently better than `BioVec` while `BioVec` performs slightly better in terms of sensitivity than `Seq2Vec`.

3.4.2 Multiclass Classification

The very high values for metrics as reported in Table 3.1 for the binary classification task is possibly due to how the negative class is constructed—by randomly selecting sequences from all the remaining families. Since the number of possible negative sequences is very large, the randomly chosen sequences would generally be far away from the positive sequences, making the positive and negative classes far apart, resulting in a relatively easy classification problem.

To alleviate this issue with the experimental setup of [20], we adopt a different approach. To compare `Seq2Vec` with `BioVecs`, we do a multiclass classification to classify the sequences into their corresponding families. We did this experiment using the 25 biggest families consisting of more than 1000 sequences each. Then we trained a multiclass SVM classifier with a linear kernel using the one-vs-rest strategy. We set $C = 1$ for `Seq2Vec` and $C = 7.5$ for `BioVecs` based on grid search. The evaluation metrics are then computed using 10-fold cross-validation and the results are reported in Table 3.2. As shown in Table 3.2, `Seq2Vec` performs much better than `BioVec` and the improvement is 4-6% for all three metrics. Note that this experimental setting provides a much harder challenge than the earlier binary classification, and the results prove that the embeddings learned using `Seq2Vec` are superior to the embeddings learned using `BioVec` for this task. Note that the low variance in the

	Precision(%)	Sensitivity (%)	Accuracy (%)
Seq2Vec	**83.37 ± 0.052**	**81.69 ± 0.067**	**81.29 ± 0.057**
BioVec	79.01 ± 0.071	76.78 ± 0.082	76.70 ± 0.080

Table 3.2: ***Multiclass Classification: Performance of*** `Seq2Vec` ***and*** `BioVec` ***for classification of sequences from top 25 families. For each evaluation metric values and corresponding std.dev is noted.***

results from the binary and multiclass experiments provides high confidence that the proposed method will generalise well for new test samples.

Having shown the improvement provided by `Seq2Vec` generated embeddings on the protein family classification tasks, in the sections below, we analyse the quality of these embeddings based on their distribution in the vector space. We also compare the results with alignment based method - BLAST.

3.4.3 Quality of Embeddings

In an ideal scenario, distributed representations should embed sequences in such a manner that different families are separated from each other, and the sequences belonging to the same class (protein family) are clustered together. This provides us with a way to do family classification using a k-nearest neighbour based approach and hence evaluate which embeddings provide a better mapping of sequences in the vector space. In this experiment, we embed the training data in a vector space using either `Seq2Vec` or `BioVec`. To assign a family to a test sequence, we find its k-nearest training sequences in the vector space and predict the family using a majority vote. Here we assume that we embed both the training and test data simultaneously. If test data are not available for embedding, we can later use gradient-descent to learn the vector representation of the test sequences.

We performed this experiment for the 25 biggest families consisting of more than 1000 sequences each, reporting results based on 10-fold cross-validation in Fig. 3.4. The results show that classification accuracy reduces rapidly for `BioVec` as compared to `Seq2Vec` with increasing neighbourhood size. This observation indicates that `Seq2Vec` embeddings for the same family are clustered closely as compared to `BioVec` in the vector space. From these results,

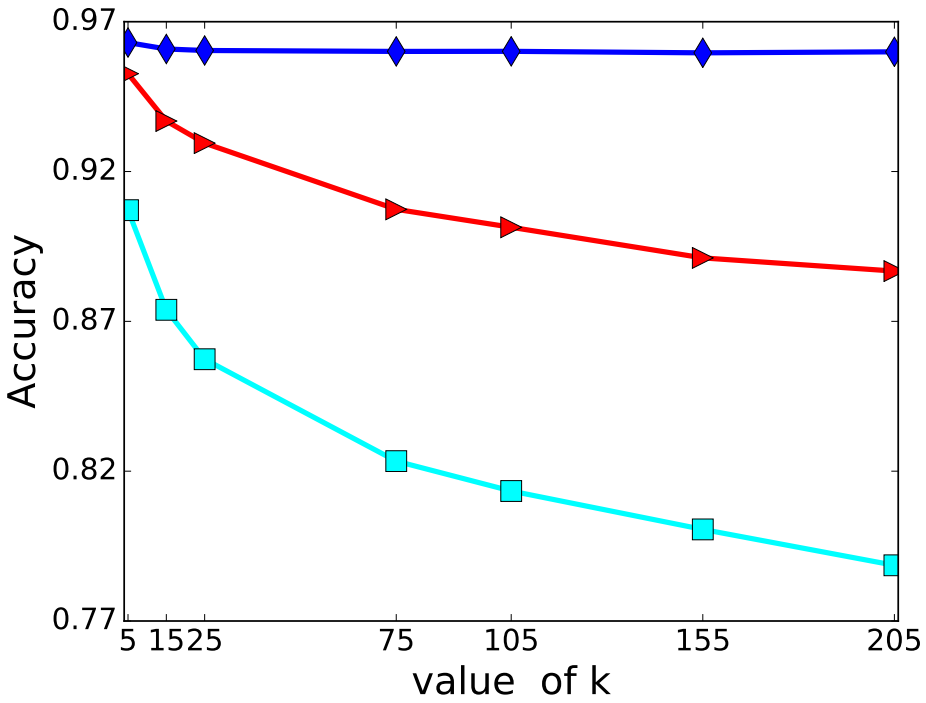


Figure 3.4: ***kNN Classification: Performance of*** `Seq2Vec` ***(–▶–▶–) and*** *BioVecs* ***(–■–■–) as compared to that of*** *BLAST* ***(–♦–♦–) as a function of*** *k* ***used for kNN.***

we can infer that `Seq2Vec` embeddings also capture the family information to some extent, hence providing better embeddings.

3.4.4 Comparison with BLAST

The BLAST heuristic finds the most similar sequences to a given query sequence based on a hit extension strategy. BLAST ranks match to a query sequence according to an estimate of the statistical significance of the match and reports the base identity via a bit score. This requires the processing of actual sequences to compute similarity, while we represent these sequences in a low-dimensional space using `Seq2Vec`. Since BLAST is a retrieval tool, to do classification, we look at the top-*k* results returned and use majority voting to assign the query to a protein family. Such setup is similar to the kNN classification setup in section 3.4.3 and hence can be compared directly. The results are reported in Fig. 3.4.

It is clear that while `Seq2Vec` managed to perform significantly better than

BioVec, BLAST outperforms Seq2Vec, specially for increasing neighbourhood size. We note that the scoring inherent in BLAST closely follows the evolutionary relationships observed across large numbers of protein alignments, and on which protein families are defined. An important focus for future work is thus to explore the extent to which this information can be captured implicitly in the context modelling of our approach, and the performance of embedding based approaches improved to become genuinely competitive with BLAST.

3.5 Conclusion

This chapter described our proposed representation learning framework Seq2Vec for biological sequences. We demonstrated the advantage of our approach by utilising the learned embeddings for protein classification problem. We showed that Seq2Vec performs significantly better than the competitive method BioVec. We also analysed the distribution of sequence embeddings in the vector space for BioVec and Seq2Vec using the kNN classification results. This analysis shows that Seq2Vec generated embeddings capture some aspects of the family information and also mapped the sequences from the same family in closer proximity when compared to the BioVec generated embeddings. Broadly, our approach falls in the alignment-free category; we therefore find it is of interest to compare it with alignment-based method - BLAST - to gain more insight. Analysing the comparison results, we see that for small neighbourhood size, the performance of Seq2Vec is similar to BLAST but the gap widens when we choose a larger neighbourhood size for making the prediction. This can be addressed by embeddings tailored to capture the other meta-information such as functional, family or structural information along with the contextual information.

Building on the insights gained from the work discussed in this chapter, we focus on proposing new representation learning approaches to improve the sequence embeddings. In the next chapter, we discuss two novel approaches for learning sequence representations - SuperVec and SuperVecX. These new embedding frameworks improve the sequence representation by incorporating available meta-information (here family information) in their learning mechanism. We demonstrate the superiority of the representations learned through

these approaches over `Seq2Vec` and `BioVec` on different classification tasks.

4 Enriching Sequence representation using meta information

Overview

In the previous chapter, we explored the use of representation learning approaches for constructing embeddings for biological sequences. In this chapter, we examine approaches to enrich the sequence representations by taking advantage of meta-information which is available in the case of many biological sequences. In contrast to the unsupervised approaches in the previous chapter, we here consider supervised learning approaches for constructing sequence embeddings. We hypothesise that including meta information - such as functional labels - along with the sequence information in the learning mechanism may enhance the quality of sequence embeddings for the particular task. Working on this hypothesis, in this chapter, we introduce two novel representation learning frameworks - `SuperVec` and `SuperVecX`. These frameworks provide a way to incorporate contextual as well as associated meta-information in their learning mechanism. Note that only training is supervised in these models, whereas the inference step (generating sequence embeddings) is unsupervised and does not require meta-information. To test the quality of the embeddings generated through `SuperVec` and `SuperVecX` frameworks, we compare them with a naive bag-of-words based representation and other established embedding techniques on various protein classification tasks. For ease of understanding, we use the same notations that were introduced in the previous chapter unless stated otherwise. The organisation of this chapter is as follows.

We begin in section 4.1 by providing a detailed discussion on available meta-informations in the bioinformatics domain and their potential use for improving sequence representations.

Section 4.2 is devoted to the description of our first proposed approach for enriching sequence embeddings : `SuperVec`. The training of `SuperVec` involves minimization of an optimization problem - discussed in section 4.2.1. Once trained, the model can be used to generate embeddings for sequences - also called as inference step (section 4.2.2). The discussion of training and inference step for our second proposed approach - `SuperVecX` is provided in section 4.3.

Having discussed details of the proposed models, next, we demonstrated the utility of sequence embeddings generated through these models on various classification tasks namely, (i) Toxin Prediction, (ii) Subcellular localization prediction and (iii) Enzyme prediction. Section 4.4 covers the description of these problems and the associated data (section 4.4.1), and the evaluation process followed for the experiments (section 4.4.2). We provide the experimental results - comparing our proposed approaches with the benchmark methods in section 4.5.

Finally, we conclude the chapter in section 4.6.

4.1 Meta Information and Representation Learning

In the previous chapter, we relied solely on the sequence or primary structure for biological entities, but in addition to the sequence, a variety of meta-information associated with these entities in many cases is now readily available. Such information includes in many cases the functional annotation of genes and their products [86], structure and domain-based categorization of proteins [87, 88] and many others. Since sequences are a limited representation of biological entities, biological inference based on sequence similarity is not always accurate. Such inference can be further improved by exploiting available meta-information along with the sequences, however this remains a challenging task.

This chapter provides ways to learn sequence embeddings by incorporating both meta-information and the contextual information present in the sequences. We propose two frameworks - `SuperVec` and `SuperVecX` in this regard.

These methods concern learning of tailored word-based embeddings for molecular sequences to support sequence comparison without the need for sequence alignment. As with other embedding methods, our techniques also require as input a set of words or *k-mers* extracted from the sequence. The effect of learning is to capture the information implicit in these *k-mers*, markedly reducing the dimension from the full bag-of-words representation while ensuring that related *k-mer* groups have proximal representations in the embedding space. We improve these associations by the addition of class information through supervision, and subsequently through the use of supervision over defined class hierarchies. In this way, we may compute sequence similarity accurately over vectors within a lower dimensional subspace, while ensuring that the calculation relies on features pertinent to the problem at hand, here reflecting some biological grouping or functional relationship.

In the earlier bioinformatics applications ([20],[68]), *k-mers* are embedded in a manner that reinforces their surrounding context, implicitly capturing the 'semantics' of these co-occurring terms. Here we implicitly adopt a distributional hypothesis for sequence *k-mers*, the view that commonly occurring *k-mers* are likely drawn from functionally related contexts, as part of some region of a gene or protein, or as a constituent of some regulatory region. The nature of these relationships depends crucially on the type of the sequence and on the resolution implied by the choice of *k-mer* size, k. The addition of metadata to the training process both reinforces and complements the relationships inherent in the sequence representation. Such labels are commonly used to indicate function (as in the *Gene Ontology* categories [86]) or associations and may be used directly (see for example, the protein-protein interaction work of [89]) or indirectly to support inference. Class information of this nature may be available implicitly in features or representations exploited by other methods, for example in the BLOSUM scores employed by BLAST [25] and MMseqs2 [30], which summarise the alignments of hundreds of proteins.

Our proposed supervised approaches capture meta information along with

the sequence content, utilising both sources to generate biological sequence embeddings. These approaches – SuperVec and SuperVecX – are inspired by the text embedding methods Word2Vec [18] and fastText [90]. For convenience we will refer to SuperVec and SuperVecX jointly as SuperVec(X). These methods make supervised learning of sequence representations based on explicit class information, extending the earlier context-based ([20], [68]) approaches to incorporate additional labels or metadata associated with the sequences – and with the resulting *k-mer* set. Our hypothesis is that by infusing meta information in the learning process, the approach will yield sequence representations better suited to the task at hand, with members of the same class (even if they are divergent sequences – those who might otherwise exhibit a lower degree of shared context) – embedded proximally within the vector space. In SuperVec, this is achieved by joining two embedding models in the one framework. Here, the first enforces class supervision, while the second incorporates contextual information present in the sequence, achieved as before by extracting a set of *k-mers* from each sequence. Class information constrains the intra-class vectors to fall closer together within the vector space, which in turn induces class information in the embedding of the constituent *k-mer* sets.

SuperVecX incorporates class-label information in the *k-mer* embeddings, using the *k-mers* of a sequence to predict the corresponding class label. The architecture is a simple linear neural network classifier that has a linear hidden layer and a softmax layer at the output. Once trained, the embedding for a new sequence is obtained by averaging the vectors of its constituent *k-mers*. Because of its simple architecture, the approach is fast and scales effectively to large sets of sequences.

Although both SuperVec and SuperVecX are supervised techniques, there are some underlying differences between them:

- SuperVec uses both contextual information as well as class-label information to learn sequence embeddings, whereas SuperVecX relies only on the class-label information. As we will show later, each of these methods may outperform the other depending upon the task at hand.
- The architecture of SuperVec is flexible and we can easily accommodate

unlabelled sequences for training – unlike SuperVecX where the class label information is essential.

In the sections below we provide descriptions of SuperVec and SuperVecX in detail. These models operate in two stages: (i) training - in this stage, parameters of the model are learned (ii) inference - learned parameters are used to generate embeddings for sequences. We describe these stages for both of the models in turn.

4.2 Proposed Approach1: SuperVec

The SuperVec architecture consists of two Word2Vec units combined so that it can jointly incorporate the class label and the contextual information contained in the sequences. As shown in Fig 4.1, NN1 is a CBOW configuration of Word2Vec that generates an embedding for sequences using contextual information, while NN2 follows a skip-gram configuration of Word2Vec to help incorporate the class level information. NN1 requires a word-context pair for training while NN2 uses a sequence and a set of similarly labeled sequences.

Note that NN1 forces the embeddings of the words that co-occur along with the sequence embedding, while NN2 further constrains sequences having the same labels to be close to one another. Achieving embeddings informed by context *and* class information requires the use of *both* networks. Here we couple NN1 and NN2 by sharing the sequence embedding between them.

4.2.1 Optimization problem formulation

To train the SuperVec model, i.e., to learn its parameters, two prediction tasks are performed, one by each of its sub-networks. Both networks essentially perform word(context)/context(word) prediction, although the meaning of context and word differs between them, as seen in Table 4.1 below. Here we see the sample *k-mer* k_{i_j}, its context C_{i_j}, the corresponding class/family label l_k and the corresponding sequence tag s_i, and the relationships between them. Since the sequence embedding task is shared by the sub-networks, the parameters of these networks influence each other. Mathematically, the coupling of these sub-networks means solving a joint optimization problem

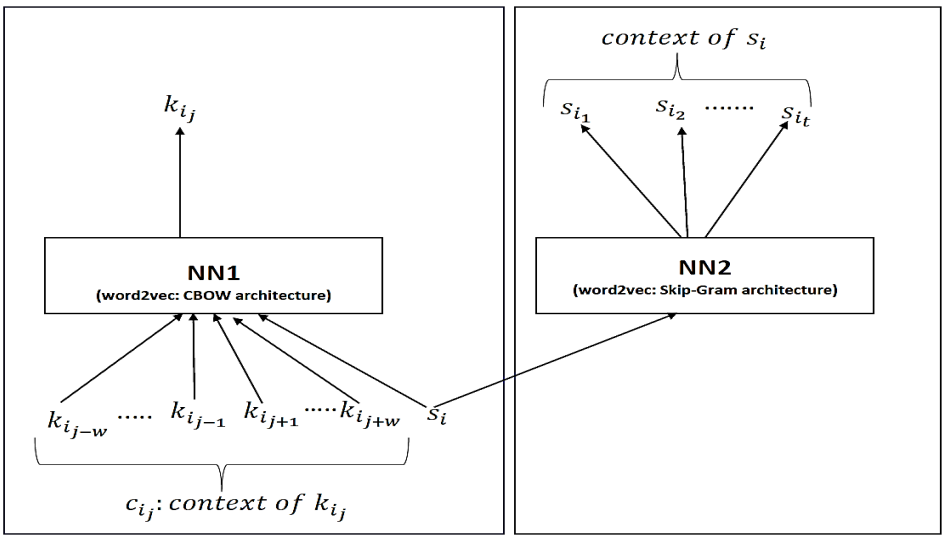


Figure 4.1: `SuperVec` **Model*: NN1 and NN2 architectures are shallow neural networks with respectively the CBOW and the Skip-gram variant of the*** `Word2Vec` ***model. The context of*** k_{i_j} ***comprises its nearby words and is denoted*** $k_{i_{j-w}}$, ..., $k_{i_{j-1}}$, $k_{i_{j+1}}$....$k_{i_{j+w}}$. ***The context of the*** s_i ***are sequences which have the same label as*** s_i ***i.e. all of these sequences belong to same family.***

Table 4.1: **Context-word pairs for NN1 and NN2**

Network	Architecture	context	word
NN1	CBOW	s_i and C_{i_j}	k_{i_j}
NN2	Skip-Gram	$s_{i_1}, s_{i_2}...s_{i_t} \in l_k$; given $s_i \in l_k$	s_i

with the overall loss function a linear combination of those for NN1 and NN2:

$$\mathrm{J}(\mathbf{S},\mathbf{K}) = \sum_{s_i \in \mathcal{S}} \sum_{j=1}^{n_i} \left(\underbrace{-\log \Pr\left[k_{i_j} \mid C_{i_j},\, s_i\right]}_{NN1} - \gamma \sum_{z \in \mathcal{I}_i^+} \underbrace{\log \Pr\left[s_z \mid s_i\right]}_{NN2} \right). \quad (4.1)$$

Here γ controls the balance between class information and contextual information. The matrices $\mathbf{K} = [\mathbf{k}_1,\ \mathbf{k}_2, \ldots, \mathbf{k}_M]^T \in \mathbb{R}^{M \times n}$ and $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2,\ \ldots, \mathbf{s}_N]^T \in \mathbb{R}^{N \times n}$ denote the embeddings for sequences and *k-mers* respectively. We define $\mathcal{I}_i^+$ to be the set of indices of sequences which have the same label as s_i. The conditional probability for NN1 in Eq. 4.1 can be computed by a softmax function, but considering the large number of *k-mers* and the computational burden involved, as before we approximate it using hierarchical softmax (HS) [77]. To compute the second part of Eq. 4.1, we use *negative*

sampling [62] where for any given sequence we try to maximize the probability of some selected positive samples as opposed to others. For example in a context/word pair, all the words in the context can be treated as positive samples and the remaining words in the vocabulary as negative samples. In our case, for any input sequence s_i, sequences whose index lies in $\mathcal{I}_i^+$ are seen as positive samples, whereas the remaining sequences are seen as negative samples. We denote the set of indices of the negative samples as $\mathcal{I}_i^-$. The second part of Eq. 4.1 can then be approximated as:

$$\log \sigma\left(\langle \mathbf{s}_z, \mathbf{s}_i \rangle\right) + \sum_{r \in \mathcal{I}_i^-} \log \sigma\left(-\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right), \tag{4.2}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function as previously defined. Maximizing the first part of the Eq. 4.2 maximizes $\langle \mathbf{s}_z, \mathbf{s}_i \rangle$ (dot product) and therefore reduces their cosine distance in the embedding space. Similarly, maximizing $\sigma\left(-\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right)$ translates into maximizing the cosine distance between $\mathbf{s}_i$ and $\mathbf{s}_r$. As defined above, the s_z have the same label as s_i whereas the s_r have a different label. Maximizing Eq. 4.2 therefore forces the sequences from the same class to be mapped closer in the embedding space, while other sequence pairs are pushed apart. Employing negative sampling thus yields embeddings with low intra-class and high inter-class separation, a mapping well suited to the retrieval task. Replacing the second part of Eq. 4.1 with its *negative sampling* expansion, the final loss function for `SuperVec` is given as

$$\begin{aligned} \mathrm{J}(\mathbf{S}, \mathbf{K}) = \sum_{s_i \in \mathcal{S}} \sum_{j=1}^{n_i} \Bigg[& -\log \Pr\left[k_{i_j} \mid C_{i_j},\ s_i\right] \\ & -\gamma \sum_{z \in Z_i^+} \left(\log \sigma\left(\langle \mathbf{s}_z,\ \mathbf{s}_i \rangle\right) + \sum_{r \in Z_i^-} \log \sigma\left(-\langle \mathbf{s}_r,\ \mathbf{s}_i \rangle\right) \right) \Bigg]. \end{aligned} \tag{4.3}$$

Computing the second part of Eq. 4.3 requires a very large number of operations—the *number of positive samples* $\times$ *number of negative samples*—for each case. To mitigate this computational burden, we use only a few randomly selected positive and negative samples for any given sequence. In Eq. 4.3, Z_i^+

and Z_i^- are sets consisting of the indices randomly chosen from the positive indices, $\mathcal{I}_i^+$, and from the negative indices, $\mathcal{I}_i^-$, respectively. Note that we sample positive as well as negative samples to further speed up the training process.

Parameter learning

The parameters of `SuperVec` govern the construction of the embeddings of the *k-mers* and the sequences. The process of learning the parameters involves a training process similar to that used for the `Word2Vec` framework. The parameters are initialized randomly and then modified for each sample to reduce the value of the loss function. The sample here consists of the *k-mer*, its context, and the corresponding sequence tag. As explained before, `SuperVec` employs two prediction tasks for each selected sample. In these prediction tasks, for any context/word at the input, the parameters are modified to maximize the probability of word/context at the output by updating the parameters values using gradient descent. After sufficient iterations, we obtain a trained network which retains the relevant information – in this case, the contextual and class information. The update equations and the derivation of the gradient for $\mathbf{s}_i$, $\mathbf{k}_{i_j}$ and other parameters of `SuperVec` over $J(\mathbf{S}, \mathbf{K})$ are provided in Appendix B. Once the model is trained it can be employed in learning the representation of any new sequence. Note that new sequences do not require any label information: as discussed below, we only use NN1 for learning this representation.

4.2.2 Inference

Computing the representation of a new sequence by passing it through the trained model constitutes the inference step. Unlike the training of `SuperVec`, since its inference step does not utilize the sequence labels, only one of the sub-networks of `SuperVec`, i.e. NN1 (refer to Fig. 4.1), is used. Although the standalone NN1 only uses contextual information, we expect that since NN1 and NN2 are coupled and jointly trained, NN1 will also capture the class information to some extent, and that this will eventually influence the representation learned in the inference step. While the representation for a sequence is computed during the inference step, all the parameters of NN1

remain unchanged except for the sequence vector, which is initialized with zeros. This vector is updated iteratively following a gradient descent approach similar to the training stage.

4.3 Proposed Approach2: SuperVecX

SuperVecX is inspired by a recent document classification method called fastText [90] that can also be used to learn sentence embedding. Unlike unsupervised word vectors, fastText provides word embeddings that can be averaged to obtain useful sentence representation. Since fastText is a linear model, training is substantially faster than the other deep learning approaches used for sentence classification. The computational efficiency offered by fastText is particularly attractive for applications in bioinformatics, which may involve millions of sequences in the training set. Below we briefly discuss the SuperVecX architecture, its parameter learning and the inference procedure.

4.3.1 Model Description and parameter learning

The architecture of SuperVecX is shown in Fig. 4.2. It is a shallow neural network consisting of an input, a linear hidden layer, and an output layer. The model is trained by presenting a sequence from the corpus to the network to predict its class.

Note that the sequence is given as a list of its sub-sequences (*k-mers*). The input to the model is obtained by averaging the vectors corresponding to the *k-mers* in the input list. These vectors constitute a *k-mer* matrix that is randomly initialized at the start of training. At the output layer, the softmax function is applied to get the probability distribution of all the classes for the given input sequence.

The overall objective of the training process is to maximize the probability (or minimize the negative log likelihood) of the classes given their corresponding sequences, hence learning the model parameters (the *k-mer* vectors and the weight matrix). Mathematically, the overall loss function is given as:

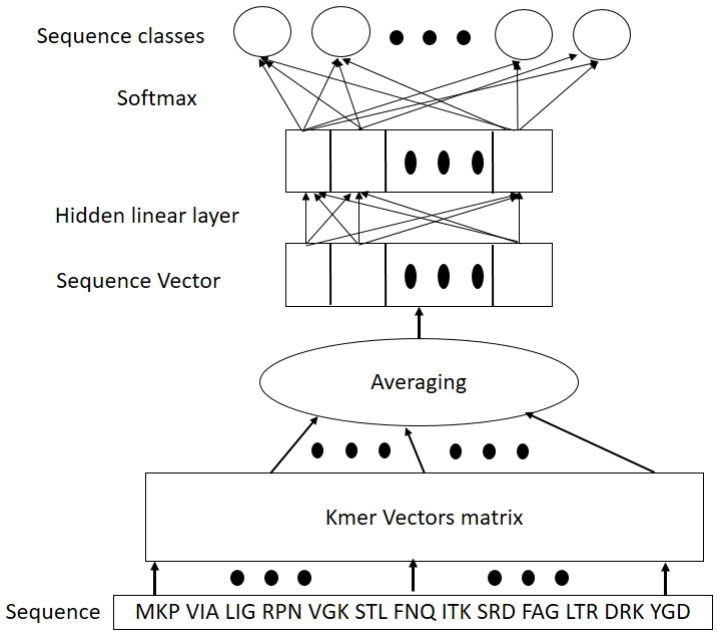


Figure 4.2: ***SuperVecX: A supervised method for generating sequence embeddings [91].***

$$\mathrm{J}(\mathbf{K}) = -\frac{1}{N}\sum_{s_i \in \mathcal{S}} l_i \log(f(Hx_i)). \tag{4.4}$$

Here, l_i is the label of s_i and H is the hidden layer matrix. The vector x_i is the average of its constituent *k-mers* as defined below:

$$x_i = \frac{1}{n_i}\sum_{j=1}^{n_i} k_{i_j}. \tag{4.5}$$

Inference

The inference step of `SuperVecX` is simple as it requires only the embeddings of the *k-mers*. Once the model is trained the embedding for a new sequence is obtained by averaging the embeddings of its constituent *k-mers*. The efficacy of the inference step is evaluated with respect to the task for which the

learned representations are employed. Here it is expected that the information learned by the trained model will be exploited in the representation of the new sequence. The flexibility of `SuperVec` and `SuperVecX` allows them to be used for different downstream bioinformatics tasks. We applied these models for sequence retrieval and classification task. We consider the application of `SuperVec(X)` to the retrieval problem extensively in the next chapter. In the section below, we consider various protein classification tasks as example downstream applications for these models.

4.4 Experimental Setup

In this section, we demonstrate the effectiveness of `SuperVec` and `SuperVecX` generated embeddings for (i) Toxin prediction (ii) Sub-cellular localization and (iii) enzyme classification tasks. Here Toxin and enzyme prediction tasks are binary classification problems whereas the subcellular localization is a multi-class (4-class) classification problem.

4.4.1 Data

We use the toxin prediction dataset provided by ToxinClassifier [92]. It contains 8093 protein sequences annotated as animal toxins and venoms. For the negative set, they adopted different sampling strategy and categorised negative sets as 'Easy', 'Moderate' and 'Hard dataset'; for our experiments we choose 'Hard dataset' that contains 7043 protein sequences.

For sub-cellular localization we use the TargetP 4-class dataset [93]. This dataset contains (i) 371 mitochondrial (ii) 715 pathway (iii) 1214 nuclear and (iv) 438 cytosolic protein sequences.

For the enzyme prediction problem, we pick the dataset provided by the 'NEW' dataset of Deepre[94]. It contains 22,168 protein sequences for the enzyme and non-enzyme classes.

4.4.2 Experimental Design and Evaluation

Classification results are computed for a 5-fold cross validation setup. We report macro-precision, recall, and F1 score as the evaluation metric. The

macro averaging is done by computing the metrics for each class and finally averaging them. Such averaging gives equal importance to all the classes. The F1 computes the harmonic mean of precision and recall values and hence captures the trade off between them.

For classification, we use a Multilayer Perceptron (MLP) classifier - which gave best results with - 3 hidden layers (70,50,20) and Rectified Linear Unit (ReLU) activation function. At the output of MLP, the softmax function is used to compute the posterior probabilities associated with each class. For comparison, we use the results provided by [95] where the authors performed the same classification tasks and reported the results for the baseline unsupervised representations - `3-mer`, `ProtVec`, `ProtVecX` and their combinations. Here, `ProtVec/BioVec` and `ProtVecX` are the unsupervised sequence embedding techniques proposed by Asgari et al. in [20] and [95] respectively. The `3-mer` representation of a sequence is obtained by normalizing its bag-of-word vector representation. The length of a `3-mer` vector is equal to $20^3 = 8000$, where 20 corresponds to number of amino acids in the alphabet.

4.5 Results and Discussion

Protein classification results using MLP classifier on toxin prediction, sub-cellular localization, and enzyme prediction is provided in Table 4.2. For sub-cellular localization and the enzyme prediction tasks, both `SuperVec` and `SuperVecX` outperform unsupervised embeddings, `ProtVec` and `ProtVecX`. For the toxin prediction task, `SuperVecX` performs at par with the other embeddings but disappointingly `SuperVec` does not perform at par with others. The results reported in Table 4.2 for unsupervised embeddings i.e. `3-mer`, `ProtVec`, `ProtVecX` and their combination are directly obtained from [95]; the size of these embeddings is 8000, 500 and 500 respectively, whereas the `SuperVec` and `SuperVecX` embeddings are of dimension 100. It is important to emphasize here that the embeddings generated through our proposed approaches are much smaller than the benchmark methods, but they still outperform these benchmarks for two of the three tasks.

Note that for any specific classification task, the `SuperVec(X)` models are

Table 4.2: **Classification results** ***: Comparing `SuperVec`, `SuperVecX` and other embeddings on various classification tasks. For each evaluation metric values and corresponding std.dev is noted.***

Dataset	Representation	5-fold cross validation results		
		macro-precision	macro-recall	macro-F1
Toxin prediction	3-mer	0.89	0.89	0.89
	ProtVec	0.88	0.88	0.88
	ProtVecX	0.88	0.88	0.88
	3-mer+ProtVec	0.90	0.89	0.89
	3-mer+ProtVecX	0.90	0.90	0.90
	SuperVec	0.86 ± 0.01	0.86 ± 0.01	0.86 ± 0.01
	SuperVecX	0.90 ± 0.01	0.90 ± 0.01	0.90 ± 0.01
Subcellular location prediction	3-mer	0.65	0.59	0.60
	ProtVec	0.60	0.57	0.58
	ProtVecX	0.57	0.57	0.57
	3-mer+ProtVec	0.68	0.60	0.62
	3-mer+ProtVecX	0.66	0.60	0.61
	SuperVec	0.63 ± 0.02	0.60 ± 0.03	0.61 ± 0.02
	SuperVecX	0.61 ± 0.01	0.57 ± 0.02	0.58 ± 0.01
Enzyme prediction	3-mer	0.70	0.73	0.71
	ProtVec	0.68	0.70	0.69
	ProtVecX	0.69	0.71	0.70
	3-mer+ProtVec	0.70	0.73	0.71
	3-mer+ProtVecX	0.71	0.73	0.72
	SuperVec	0.74 ± 0.001	0.74± 0.001	0.74± 0.001
	SuperVecX	0.73± 0.001	0.73± 0.002	0.73± 0.001

trained on the protein sequences available for that task. The better performance of supervised embeddings on classification tasks shows that the information present in the sequences can be well captured in a low dimensional vector. As noted before, `SuperVec` relies on contextual and class information whereas `SuperVecX` utilizes only class labels. The difference in the performance of `SuperVec` and `SuperVecX` over these tasks suggest that for some tasks class label information may be sufficient, but for others contextual information may further improve the embeddings.

4.6 Conclusion

In this chapter, we introduced two new approaches – SuperVec and SuperVecX for learning embeddings of biological sequences. These approaches provide a way to enhance the quality of sequence embeddings by incorporating meta-information (labels) along with the information contained in the sequences themselves. The SuperVec model incorporates both contextual and sequence labels in its learning mechanism, whereas, SuperVecX relies mainly on the sequence labels. We demonstrate the applications of these embeddings on different bioinformatics problems, namely, (i) Toxin prediction (ii) Sub-cellular localization and (iii) enzyme classification. The results obtained on these tasks show that using meta-information can improve the quality of embeddings and hence providing results that are leastwise at par with the embeddings that are of much higher dimension.

The flexibility of these approaches make them ideal candidates for other bioinformatics tasks, especially where a large number of sequences need to be compared, such as database search/sequence retrieval. In such cases, higher-dimensional representations of sequences may pose memory and computational challenges, hence lower-dimensional representations of sequences may prove to be more usable.

Considering these potential benefits, in the next chapter, we explore the utility of learned representations for sequence retrieval task. Conventionally, this area is dominated by alignment-based techniques such as BLAST and MMseqs2. As discussed earlier in section 2.1, alignment-based approaches inherit some limitations, including their inability to scale well with increasing database sizes. Working with low dimensional sequence embeddings is expected to address some of these limitations. In the next chapter, we first establish the superiority of our proposed supervised representation learning approaches over their unsupervised counterpart for the homologous retrieval task. We also introduce a new hierarchical framework (build on SuperVec(X) models), and a hybrid approach, SuperVec(X)+BLAST specifically for this task. Experimental results show that our proposed approaches provide comparable results to the alignment-based techniques while giving an order of magnitude improvement in querying time.

5 Using Sequence Embeddings for Sequence Retrieval

Overview

Having introduced new approaches for learning sequence representations and refining them to add supervision in chapter 4, in this chapter, we explore the utility of embeddings for the homologous sequence retrieval task. This task is computationally expensive as it requires a large number of sequence comparisons. Using low-dimensional embeddings of sequences for such a problem will potentially provide computational advantages. Although this area is dominated by alignment-based methods such as BLAST, in this work, we propose competitive, embedding-based (alignment-free), and hybrid (alignment+alignment-free) approaches. The experimental results demonstrate that our proposed approaches can achieve performance equivalent to the alignment-based methods while providing an order of magnitude improvement in the querying time. The organisation of this chapter is as follows.

We briefly discuss the homologous sequence retrieval task, related computational issues and our proposed approaches in section 5.1. We provide details of the data used for experiments in section 5.2. After that in section 5.3, we provide details of experimental setup and evaluation metrics. Following on, in section 5.4, we provide a comparison of different representation learning techniques on the retrieval task. Taking hints from the experimental work discussed in section 5.4, we propose an approach, `H-SuperVec(X)`, specifically designed for the sequence retrieval task. Details of `H-SuperVec(X)` are provided in section 5.5. The improvement in retrieval performance provided by `H-SuperVec(X)` when compared to other approaches is discussed in

section 5.6. Apart from the accuracy, querying time is an important metric for judging the retrieval techniques. In section 5.7, we provide a comparison of querying time for all of the methods considered, including standard alignment-based approaches - BLAST and MMseqs2.

We note that while the embedding-based approaches give an order of magnitude improvement in querying time, there remains significant scope for improvement in the retrieval performance. We therefore seek to improve the retrieval accuracy while maintaining the computational advantages. We look at this through a hybrid combination, which we call `H-SuperVec(X)+BLAST`, details of which is provided in section 5.8. Finally, in section 5.9, we conclude this chapter.

5.1 Homologous sequence retrieval

The homologous sequence retrieval task pertains to the retrieval of sequences with a common evolutionary history from a database. Here, we take membership of the same protein family as a marker of homology, relying on the family definition provided in PFAM [87] for protein sequences. The exponential increase in the database size over the last decade or more has posed fundamental challenges for alignment-based techniques for large scale sequence comparisons. The computational requirements of these techniques increases with the size of database[1] and input sequence. Our proposed representation learning techniques (`SuperVec` and `SuperVecX`), reduce the dimension of any length sequence into a small fixed-length vector and are hence able to provide an order of magnitude improvement over alignment-based methods in execution time for the homologous sequence retrieval task. For the embedding-based methods, we address the retrieval problem by employing the nearest neighbour search, where for a given query, the sequences from a database are retrieved based on the cosine-distance of query and database sequence embeddings. The complexity of comparing two sequences in the proposed approaches is linear, i.e., $\mathcal{O}(m)$ where m is the dimension of a sequence embedding.

[1] sum of the length of all database sequences

For both proposed methods, the number of classes affects the efficacy of the training process, ultimately limiting accuracy. To overcome these issues, we consider a range of partitions other than the original classes and construct a series of embeddings using the `SuperVec(X)` algorithm to better cover the space. This method, which we call `Hierarchical SuperVec(X)` or `H-SuperVec(X)`, supports embeddings across feature spaces based on the class labels and their exclusive and exhaustive subsets. Further, these methods can also be used as a filter to rapidly select a relevant subset of sequences from a large database, and then applying methods with high precision to the subset to give the desired output. We call this strategy the Hybrid approach. The experimental results show that the Hybrid approach – `H-SuperVecX+BLAST` – is significantly faster than BLAST and gives similar performance for early recall levels for the retrieval task. In the sections below, we first provide details of the datasets used, the experimental setup and the evaluation metrics before covering the retrieval experiments in detail.

5.2 Data

We use the subset of protein sequences from the Swiss-Prot[20] and Uniref50[96] databases for the evaluation of our approaches on the sequence retrieval task. For convenience we call these datasets as Dataset1 and Dataset2 respectively. The total number of sequences in Dataset1 and Dataset2 are 150k and 1.1M respectively, as indicated as the subscript in Swiss-Prot_150k, and Uniref50_1.1M.

The description of these datasets is as follows:

Dataset1: Swiss-Prot_150k

The Swiss-Prot dataset consists of 324018 protein sequences, each uniquely annotated with one of the 7027 family labels. In this dataset, there are families which have the same functional description as in the PFAM database. For example, Chitin_synth_1 (PF01644) and Chitin_synth_2 (PF03142) are two different entries in PFAM, but both represent the chitin synthase enzyme. We merged such families into a single representative family leading to a reduction in the number of families to 6967. The families/classes present in this dataset

differ considerably in their size; the largest family contains 3024 sequences, whereas many families are based on a single sequence.

To create reasonably sized training and test splits for experiments, we chose the largest 200 families from this dataset, ensuring a minimum family size of at least 400 members, making our total dataset size 150,324 sequences.

Dataset2: Uniref50_1.1M

The Uniref50 database contains clusters with sequences that share 50% or more similarity; each cluster is represented by a consensus sequence also called the representative sequence. After pre-processing the data, we keep only those sequences that are annotated with only one "PFAM" entry. For the experiments, we keep 100 largest classes that contain $400 - 1000$ members, making our total dataset size 96,612 sequences.

A detailed description of both the datasets is given in Appendix A.

5.3 Experimental Design and Evaluation

The experimental setup used to conduct the retrieval experiments in this work is discussed below.

- **Setup 1:** This setup is designed to demonstrate the advantages of the supervised methods `SuperVec`, `SuperVecX` and their extensions `H-SuperVec` and `H-SuperVecX` over unsupervised embeddings on the sequence retrieval task. Here, the database is constructed by choosing at random 60% of the sequences from each class (each family corresponds to one class). The remaining 40% of these sequences form a query set that is used to evaluate retrieval performance on the database. Retrieval based on a given query proceeds as follows. First, for each database sequence, an n-dimensional embedding is generated, yielding a database embedding space. For a new query, we first generate its embedding; then we employ nearest-neighbor search in the database embedding space, returning sequences in descending order of cosine similarity using the nearest neighbor search [97].

Note that the process of generating embeddings for the sequences differs for each of the considered Representation Learning (RepL) methods. For `BioVec`, sequence embeddings are generated by adding the corresponding *k-mer* embeddings; we use the *k-mer* embeddings provided by [20]. To generate the sequence embedding using the `Seq2Vec` or `SuperVec` approach, we first train them using the database sequences; note that unlike `Seq2Vec`, `SuperVec` also uses the database sequence label for training. Once these models are trained, the sequence vectors are generated by the inference step. For `SuperVecX`, embedding for a new sequence is obtained by adding embeddings of its constituent *k-mers* - that are learned by training `SuperVecX`.

In this study, all experiments are performed using a commodity Linux workstation equipped with an Intel Core i7-4790K, 3.6GHz 8 core, 16 thread processor. The values of hyper-parameters–the *k*-mer size and the representation length of sequences for `SuperVec` and `Seq2Vec`–are kept same as given in [20], whereas the context size and supervision parameter γ are chosen based on the best retrieval results obtained on a database of the largest 50 classes under setup 1. We also evaluated our methods for a few different values of k (3, 4, 5) and found that retrieval performance remains almost the same. The values selected for the `SuperVec`, `SuperVecX` and `Seq2Vec` hyper-parameters are shown in Table 5.1.

Table 5.1: **Hyper-parameters for `SuperVec`, `Seq2Vec` and `SuperVecX`**

Parameter	Value
k-mer size	3
context size	1
Sequence representation length	100
γ	0.5

5.3.1 Evaluation

The performance of the supervised embedding methods and their hierarchical versions on the homologous sequence retrieval task is evaluated through comparison with the other RepL approaches – `BioVec` and `Seq2Vec` – and with standard alignment based approaches, BLAST and MMseqs2. The evaluation metrics for the retrieval task are chosen as follows:

1. **Interpolated precision-recall values:**
   Here we first provide definition of recall and precision.

   (a) **Recall**: It is a measure of the ability of a system to retrieve all the relevant items

   $$r = \frac{Rel@r}{number\ of\ relevant\ entries\ in\ the\ database}. \tag{5.1}$$

   (b) **Precision**: It is a measure of the ability of a system to retrieve only the relevant items. When calculated at recall level r, it is given as:

   $$p(r) = \frac{Rel@r}{ER_{Rel@r}}. \tag{5.2}$$

   Here, $Rel@r$ is the number of relevant entries and $ER_{Rel@r}$ represents the total number of entries retrieved at r. Recall levels are normalised values and range from 0 to 1; each level indicates the percentage of relevant entries, e.g. 0.1 recall level means 10% of relevant entries. For a query Q with label L, the relevant entries are the database sequences with label L.

   The interpolated precision value $p_{interp}(r)$ [98] at a recall level r, is defined as highest precision value found for any recall level $r^{'} > r$:

   $$p_{interp}(r) = \max_{r^{'} \geq r} p(r^{'}). \tag{5.3}$$

   We compute interpolated precision values for each query across all recall levels for the retrieval experiments, which are then averaged to give final retrieval performance. For a query Q with label L, $p(r^{'})$ in Eq. 5.3 can be written as, $p(r^{'}) = \frac{Rel@r^{'}}{ER_{Rel@r^{'}}}$. Here, $Rel@r^{'}$ is $r^{'}$% of the database sequences (entries) with label L.

   Note that the entries are sorted based on their similarity to the query sequences, usually computed based on the alignment score. For RepL approaches, we use the cosine similarity of query and the entries to compute the similarity. The $Rel@r^{'}$ is fixed for any search algorithm, a

better search algorithm has a smaller $ER_{Rel@r'}$, and hence gives higher $p(r^{'})$.

2. **Querying Time:**

   The querying time is defined as the time required to retrieve the significant matches from the database for a given query. As discussed before, the retrieval process for RepL methods requires that we first train the model, which is then used for generating the representations for the database and subsequent query sequences, followed by the nearest-neighbor search. This training process is computationally intensive, but is required only once for each model. The training time for `SuperVec` for $\backsim$ 90k training sequences is $\backsim$ 28 hours, and there remains scope to improve this performance by using a distributed training framework for our approach. The idea is to use multiple GPUs on a cluster to train in a distributed fashion and then to use a reduce operation to compute the full gradient. We expect the training time to linearly decrease as we increase the number of GPU's. Since `SuperVecX` is a linear classification model , it is sufficiently fast to work with much larger datasets; it takes around ten minutes for training over $\backsim$ 90k sequences. We compute the querying time as the sum of the time taken to generate the sequence embedding and the time required to produce the list of nearest neighbor(s) for a given query. For BLAST and MMseqs2, we seeks to report the querying time as the time required to return the list of database sequences for a given query.

In the section below, we compare the effectiveness of supervised and unsupervised embeddings on the homologous sequence retrieval task.

5.4 Supervised vs Unsupervised Embeddings

To demonstrate the effect of supervision on retrieval performance, we initially consider a retrieval task over a small, two-classes database from dataset1. The results averaged over 100 randomly chosen pairs are provided in Table 5.2. The result shows that the proposed supervised embedding approaches outperform the unsupervised approaches over this task. As the RepL approaches

fall broadly under the umbrella of alignment-free methods, we also consider the performance of BLAST - the most widely used of the alignment-based approaches - on these tasks. The results show that `SuperVec` and `SuperVecX` outperform BLAST for the two-class database problems.

Table 5.2: **Retrieval results for** 100 **random pairs***: Average interpolated precision values at ten recall levels computed for* 100 *random pair of classes. All of these pairs differ in the number of database and query sequences. The precision value shown at particular recall level below is averaged over the chosen* 100 *pairs.*

	Recall Levels				
Methods	0.1	0.2	0.3	0.4	0.5
`BioVec`	0.9079 ± 0.087	0.8817 ± 0.101	0.8573 ± 0.112	0.8329 ± 0.124	0.8066 ± 0.134
`Seq2Vec`	0.9494 ± 0.034	0.9258 ± 0.05	0.904 ± 0.064	0.8828 ± 0.076	0.8616 ± 0.085
`SuperVec`	0.9899 ± 0.009	0.9882 ± 0.011	0.9868 ± 0.013	0.9853 ± 0.014	0.9836 ± 0.016
`SuperVecX`	0.9339 ± 0.08	0.9261 ± 0.1	0.9216 ± 0.10	0.9176 ± 0.11	0.9146 ± 0.11
`BLAST`	0.9705 ± 0.046	0.9562 ± 0.082	0.9416 ± 0.105	0.9322 ± 0.115	0.9208 ± 0.133
	Recall Levels				
Methods	0.6	0.7	0.8	0.9	1
`BioVec`	0.7764 ± 0.142	0.7458 ± 0.147	0.7109 ± 0.149	0.6617 ± 0.143	0.5520 ± 0.082
`Seq2Vec`	0.8380 ± 0.094	0.8110 ± 0.102	0.7778 ± 0.108	0.7320 ± 0.111	0.6247 ± 0.095
`SuperVec`	0.9815 ± 0.018	0.9788 ± 0.021	0.9749 ± 0.025	0.9675 ± 0.032	0.9279 ± 0.06
`SuperVecX`	0.9095 ± 0.126	0.9038 ± 0.13	0.8941 ± 0.14	0.8782 ± 0.15	0.8440 ± 0.17
`BLAST`	0.9084 ± 0.153	0.8918 ± 0.180	0.8547 ± 0.214	0.7896 ± 0.259	0.0485 ± 0.136

We also visualise and compare the embeddings of the database and query sequences generated through our proposed supervised approaches and the existing unsupervised methods in two-dimensional space. We use the t-SNE [99] tool to provide the two-dimensional mapping of the embeddings. The plots provided in Fig 5.1 are obtained for largest two classes of dataset1; the enlarged figures for these plots are provided in Appendix E.

From the plots in Fig 5.1, we observe that:

- `BioVec` generated database embeddings are well-separated by class, but form small groups within each class. `Seq2Vec` provides (relatively) better intraclass organisation than `BioVec`, but the two classes are merged to a great extent. `SuperVec` database embeddings show a better intraclass and interclass separation than the other methods, albeit with some overlap at the boundary.
- For a new query, the presence of the relevant entries (here the database sequences from the same class) decreases as we increase the neighbourhood size. Thus, we expect to see a decrease in the precision values

for increasing recall levels. Analysing the plots we can infer that such an effect will have a stronger impact on the retrieval performance of `Seq2Vec` and `BioVec` when compared to `SuperVec`, especially for late recall levels. For both database and query sequences `SuperVecX` provides well-separated embeddings in the vector space, which will thus be expected to give the best result among all considered RepL methods for the chosen classes.

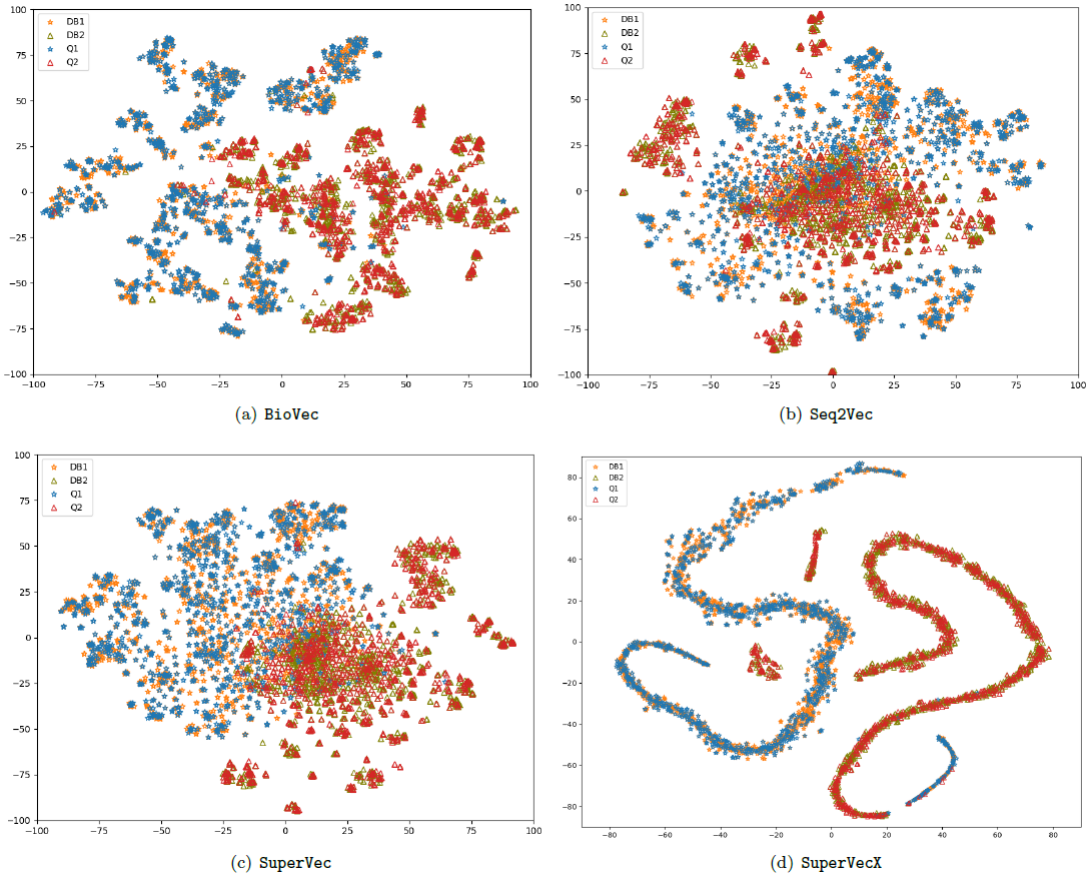


Figure 5.1: **t-SNE plots***: The mapping of database and query embeddings generated through* `BioVec`*,* `Seq2Vec` *and* `SuperVec` *and* `SuperVecX` *approaches. DB1, DB2 denotes the database sequences and Q1 and Q2 denotes the query sequences from class1 and class2 of dataset1.*

These experiments are subsequently extended to much larger databases involving a large number of classes and sequences. We again follow experimental setup 1 and limit our analysis to the 200 classes (150,324 sequences) from dataset1, ensuring a minimum size of 400 samples. We also test our proposed methods on the largest 100 classes (96,612 sequences) of dataset2. This

dataset is more challenging than dataset1 as no two sequences share more than 50% sequence similarity. The results of these experiments are provided in Fig 5.2.

Note that although we only show the results for larger databases (200 and 100 classes) in Fig 5.2, we performed the same retrieval task on the database of different sizes and find that the results remain consistent with the experiments shown.

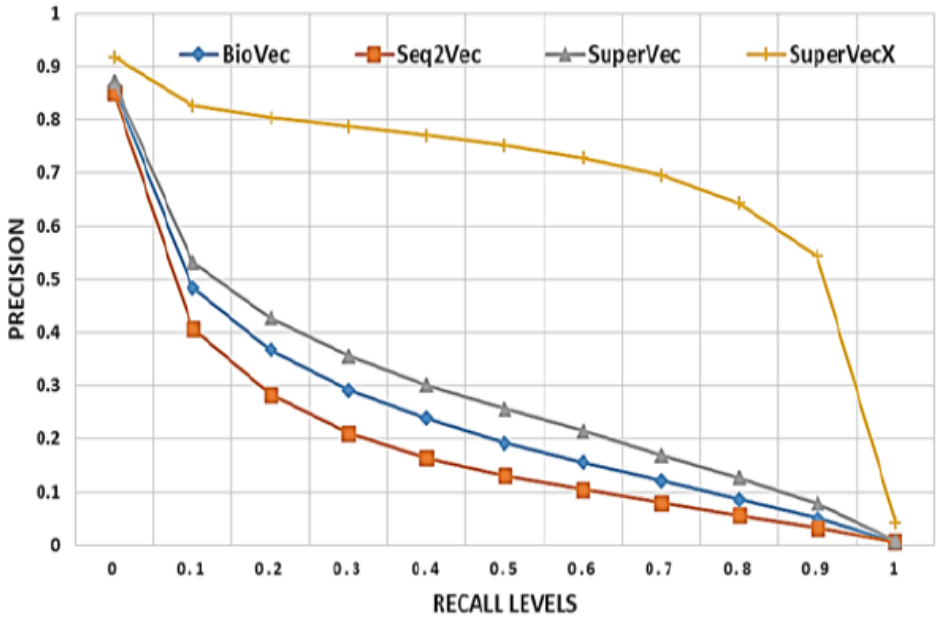
Robustness of `SuperVec` and `SuperVecX`

In setup 1, we consider the scenario when labels for all the database sequences are known; hence can be used for training purposes. Although the inference step (learning embeddings of database and query sequences) is unsupervised, utilising the database sequences for training might indicate that the model might overfit and not generalise well. To test the robustness of `SuperVec` and `SuperVecX` to the variation in selection of the training set, we experimented following setup 2 that differ in the way it split data into query, database and training set compared to setup 1 and compared their results over retrieval tasks.

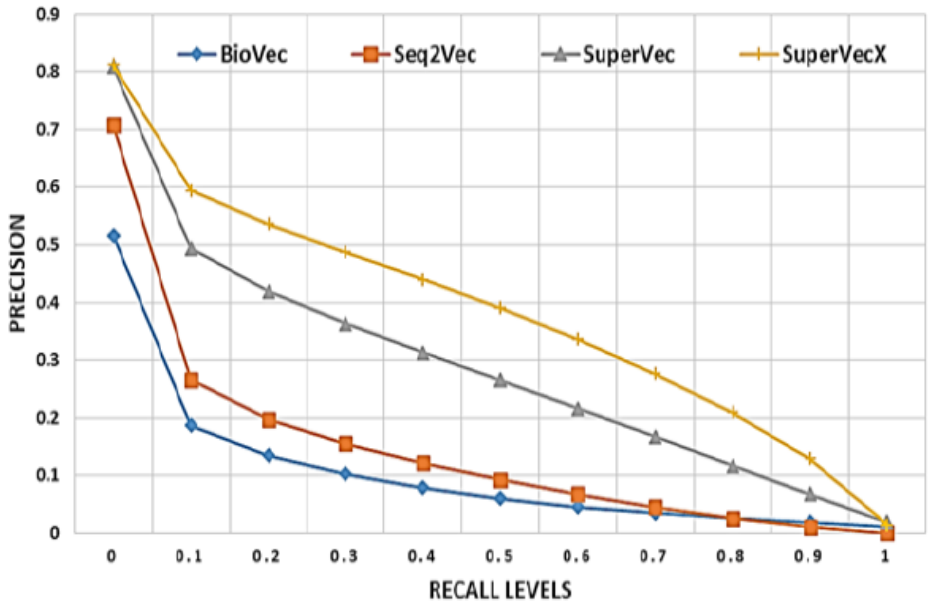
Details of setup 2 is as below.

- **Setup 2:** In this setup the sequences are chosen randomly from each class in the ratio 60:20:20, generating the training, database and query sets. Here the training sequences are used for training the models whereas the database and query sets are reserved for validation on the retrieval task. Once the models are trained, the process followed to generate sequence embeddings and retrieval of homologous sequences for a given query is the same as described in setup 1. Note that this change has little effect on `BioVec` - generated sequence embeddings as we use the *k-mer* embeddings directly from [20].

The retrieval results obtained following setup 2 (Fig 5.3) are consistent with setup 1 (Fig 5.2), where supervised approaches outperform the baseline RepL methods. These results confirm that the proposed models generalise well and are robust even when trained with sequences not present in the database.



(a) Dataset1



(b) Dataset2

Figure 5.2: **Supervised Vs unsupervised:** ***Average interpolated precision values at 11 recall levels for dataset1 and dataset2. For dataset1 the results are averaged for*** $\sim 60k$ ***queries, the database size is*** $\sim 90k$ ***(200 classes); for dataset2 the database size is*** $\sim 58k$ ***sequences (100 classes) and the results are averaged over*** $\sim 38k$ ***queries.***

As noted before for the two class experiments, SuperVecX is seen to outperform BLAST, especially at late recall levels.The retrieval result for the database holding largest 100 classes from dataset1 having 75,576 sequences in total is provided in Fig 5.3. Since the database and training sequences are different in setup 2, the superior results obtained using SuperVec(X) suggest that SuperVec(X) transfers the information learned from the training sequence and their labels efficiently to the database and query sequence embeddings.

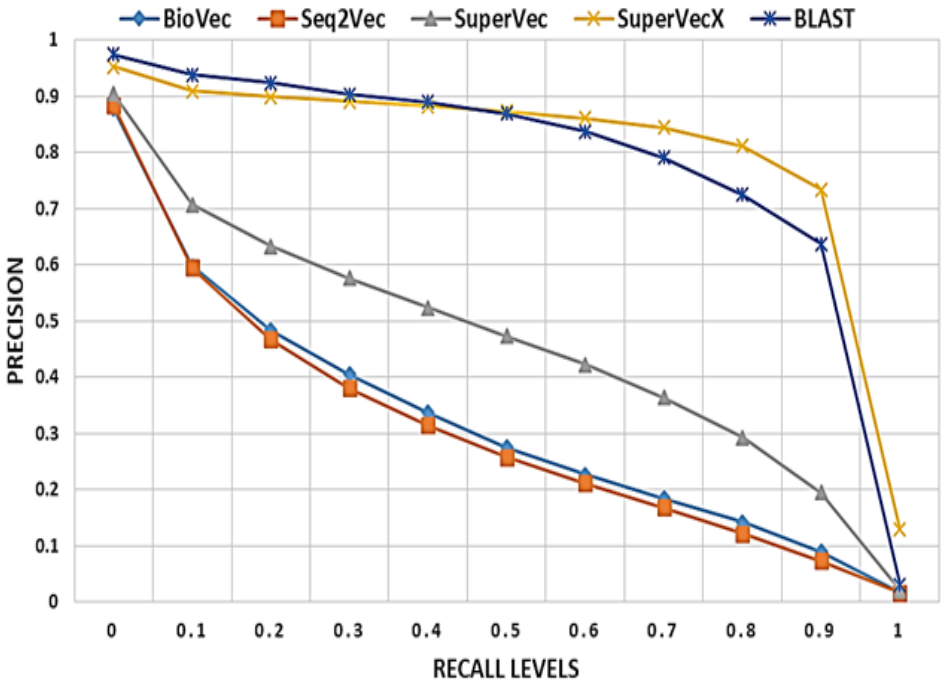


Figure 5.3: **100 classes experiment***: Comparison of interpolated average precision values for retrieval task performed on largest 100 classes database of dataset1 following setup 2.*

The comparison of unsupervised methods (Seq2Vec, BioVec) and SuperVec(X) results (see Section 5.4) for retrieval tasks confirms our hypothesis that incorporating label information in the training process helps in generating useful sequence embeddings for retrieval purposes. Although SuperVec and SuperVecX perform better than other representation learning methods (Seq2Vec, BioVec [20]) for a number of retrieval tasks, we observe that their retrieval performance deteriorates with an increasing number of classes.

For SuperVec, with the increase in the number of classes, the constraints enforcing interclass and intraclass separation necessarily increase in number.

Satisfying this set of constraints may prove difficult, reducing the efficacy of the training process and ultimately leading to a deterioration in retrieval performance. Although the interclass and intraclass constraints are not imposed explicitly for SuperVecX, these constraints are enforced implicitly by following the classification task involved in its training. The other important observation to note here is that the interclass and intraclass distances of a set of sequences change as we generate sequence embeddings using SuperVec(X) models trained on a diverse set of classes. This observation implies that we get a diverse embedding of a sequence when it is generated through multiple models.

Keeping these observations in mind, we propose an hierarchical approach that improves retrieval results by computing a better estimate of the query-target similarity. We call this proposed method H-SuperVec or H-SuperVecX for SuperVec and SuperVecX respectively, or collectively as H-SuperVec(X). In this approach, we work with a binary-tree obtained by partitioning a set of classes at each parent node (refer Fig. 5.4). Once the tree is created, the SuperVec(X) model is trained for each of these nodes. Following this approach gives us multiple trained models, which can be used to generate many observations of the same quantity (here the query-target[2] distance). These observations are processed jointly to get a better estimate of query-subject similarity.

5.5 Hierarchical SuperVec(X)

H-SuperVec(X) exploits the fact that we can generate multiple, diverse embeddings for a given sequence using multiple SuperVec(X) models trained on sequence data belonging to a diverse set of classes. Note that for each query-target (database sequence) pair, each SuperVec(X) model results in a distance computation corresponding to that model. This distance is used as a proxy to measure the similarity between two sequences. Each SuperVec(X) model introduces some noise in the embeddings it generates and hence in the distance computed for any query-target pair. Processing the query-target distances obtained from different SuperVec(X) models together reduces the

[2]target: database sequence

overall noise and gives us a better estimate of query-target similarity. We utilise this fact in `H-SuperVec(X)` and apply it to the same retrieval task. Applying `H-SuperVec(X)` for retrieval tasks involves the following steps:

- Forming an Hierarchical Structure: First, we assign all of the class labels and their corresponding sequences to the root node. The root node is then split into two child nodes by randomly partitioning its associated class labels into equal halves. These child nodes are further partitioned, following the same process for each node until we are left with leaf nodes, each associated with a single class label. An example of such an hierarchical tree is provided in Fig. 5.4.

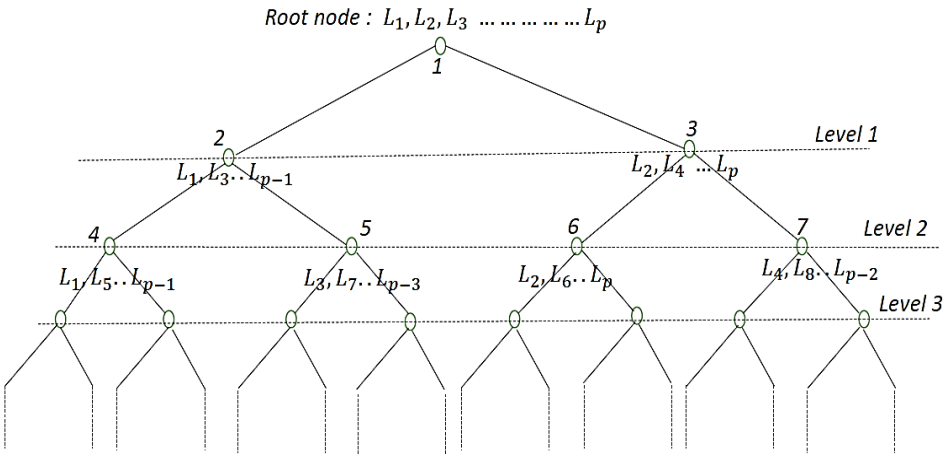


Figure 5.4: **An example Binary tree for `H-SuperVec(X)`*: A hierarchical structure obtained by partitioning the class labels of each parent node into equal size subsets. The root node is assigned p class labels ($L_1, L_2 .... L_p$) and their corresponding sequences. In this example we assume that p is an even number; the right child of the parent node is assigned the even and the left child is assigned the odd indexed labels selected from those assigned to the parent node.***

- We then train a `SuperVec(X)` model for each node of the above tree: these models can subsequently be used to generate embeddings for a new sequence.
- Assign weights to each node: As we traverse down the tree, `SuperVec(X)` is successively trained with fewer classes, leading to an increase in noise in the computed query-target distance.

  To get a better estimate of query-subject similarity, we apply a simple linear model (weighted sum) over the distances computed at each node.

For query (q), the similarity is estimated as the weighted sum:

$$d = \sum_{i \in nodes} w_i d_i, \tag{5.4}$$

where d_i is the query-subject distance computed at the i^{th} node and w_i is the weight assigned to node i. Since noise increases as we traverse down the tree, the largest weight is assigned to the root node, with node weights decreasing as we traverse toward the leaves. The weight magnitudes are constrained by three conditions: (i) the weights are positive, $w_i > 0, \forall i$; (ii) the weights at child nodes are assigned based on the probability (computed at parent node) with which a query can be assigned to the child nodes. For our experiments we use nearest neighborhood classification for assigning probabilities; (iii) the weights sum to one, i.e., $\sum_{i \in nodes} w_i = 1$.

- Retrieve sequences: Once we build a hierarchical tree and train the `SuperVec(X)` model for each node, the retrieval task is performed as follows. First, we learn embeddings for database sequences using the `SuperVec(X)` model at each node. For a new query sequence, multiple embeddings are generated corresponding to the nodes of the tree using the inference step. Pairwise query-target distances are then calculated for each node using Eq (5.5). These distances are finally combined as a linear sum to give an estimate of similarity between the query-subject pair.

  The results are returned in descending order of their similarity with the query sequence.

Since the weight assigned to each node is reduced as we traverse down the tree, the contribution of the nodes in the computation of similarity between query-subject pairs also decreases. We empirically determined that working with a one or two level tree produces consistently better retrieval results. For our experiments, we chose a tree with only one level, i.e., the tree having the root and its child nodes. We demonstrate the mechanism followed by the `H-SuperVec(X)` method to estimate the distance of the query-target pair in Fig 5.5, where `SuperVec(X)1`, `SuperVec(X)2` and `SuperVec(X)3` are the models for root and first level nodes respectively. Each of these models is utilized to

obtain the query-subject distance; the distance computed at the i^{th} node and hence through i^{th} model is given as:

$$d_i = 1 - \frac{\langle \mathbf{q}_i, \mathbf{s}_i \rangle}{\|\mathbf{q}_i\| \|\mathbf{s}_i\|}. \tag{5.5}$$

here $\mathbf{q}_i$ and $\mathbf{s}_i$ are the embeddings of query 'q' and a database sequence 's' computed at the i^{th} node.

For the experiments we fix the node weight to be 0.5; the weights for other nodes are chosen following the process as discussed before. Employing hierarchical approaches `H-SuperVec(X)` for the retrieval task gives an improvement over other representation learning methods considered in this work including the naive `SuperVec(X)` methods; the results are considered in the section below.

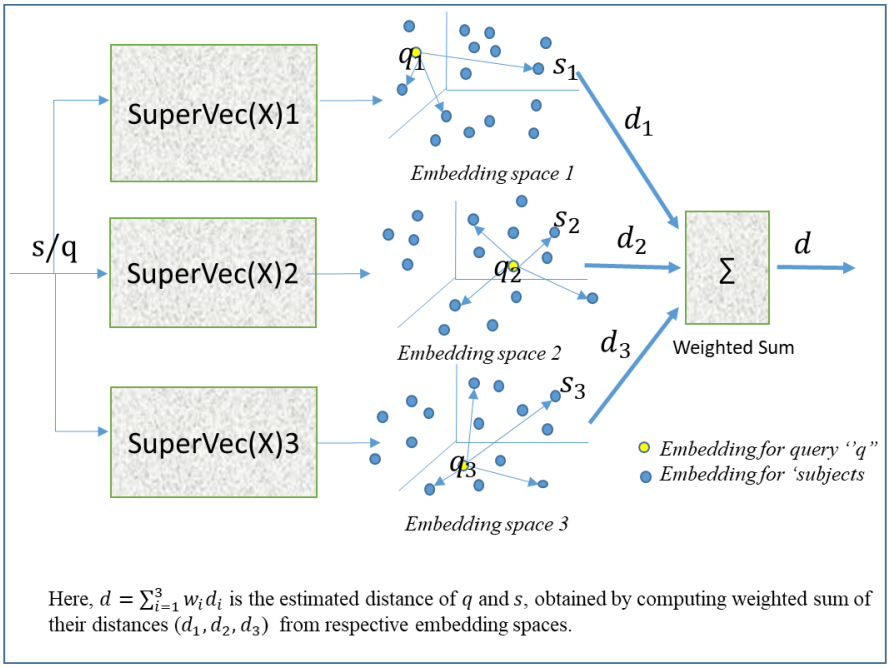


*Figure 5.5: `SuperVec(X)`**1/2/3 are the `SuperVec(X)` model trained for node 1, 2 and 3 respectively. q_i and s_i are the embedding of a query (q) and subject (s) and d_i is the distance of query-subject pair computed at i^{th} node, d is the similarity score for q and s.***

.

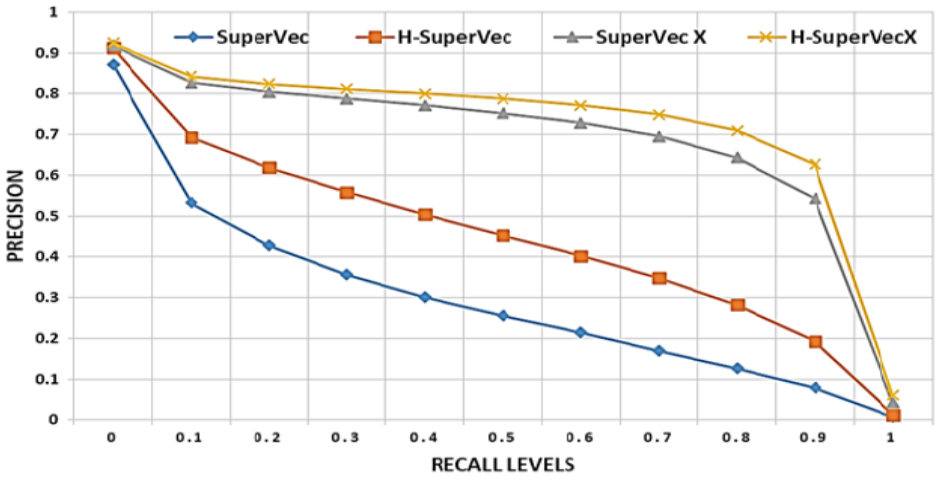
5.6 Retrieval using H-SuperVec(X)

As discussed above, the hierarchical approach is specifically designed for the sequence retrieval task, and may be used with any supervised RepL method. When used with our proposed methods – SuperVec or SuperVecX – we call these methods H-SuperVec and H-SuperVecX respectively. In this approach, the set of classes in the database is partitioned into a series of exclusive and exhaustive subsets leading to a binary tree (Fig 5.4). We then train the SuperVec(X) model for each of the subset nodes of the tree. For a given set of classes, many such partitions are possible, and the choice of partitioning may affect the training of SuperVec(X) network, which in turn affect the query-subject similarity computation and subsequent retrieval performance.

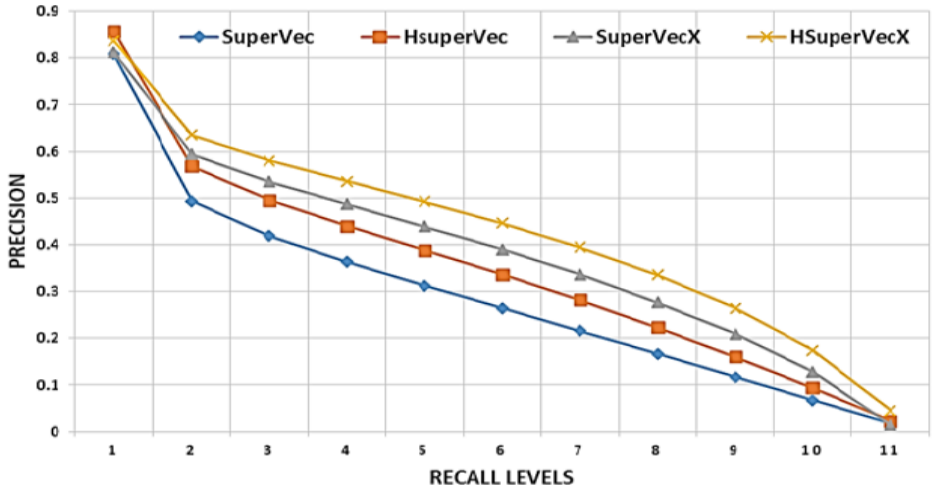
To analyze the effect of partitions on retrieval results, we perform the retrieval experiment on different splits of a randomly chosen set of eight classes from the dataset1, with the experimental protocol following setup 1. The details are provided in Appendix C. The results from this experiment show that the choice of the partition at the root node has a negligible impact on the performance of H-SuperVec; in other words, a random partition may be chosen.

Based on the observations from the eight class experiments, we use a random partition of equal size for applying the hierarchical approach on larger databases – (i) $\sim 90k$ sequences (largest 200 classes) of dataset1 and (ii) $\sim 58k$ sequences (largest 100 classes) of dataset2. It is observed that H-SuperVec provides an improvement in precision values as high as $\sim$ 70%, whereas H-SuperVecX provides improvement of $\sim$ 30% compared to SuperVec and SuperVecX respectively. The results are shown in Fig 5.6. These results validate our claim that using multiple observations of the distance give a better measure of the similarity of the sequences in the embedding space, leading to a superior performance on the retrieval task.

We now consider the querying time required for each of these methods in the section below.



(a) Dataset1



(b) Dataset2

Figure 5.6: **H-SuperVec(X) vs SuperVec(X)***: Retrieval performance comparison of the hierarchical and vanilla embedding approaches for dataset1 and dataset2.*

5.7 Querying Time

In this section, we provide a comparison of the querying time of all the methods considered for the sequence retrieval task. For the querying time computation, we use a database of $\backsim$ 90k sequences and 200 classes of dataset1 that are searched for a set of 1108 queries (1% from each class). These query sequences are selected randomly and in equal proportion from each class. We report the querying time for two possible cases: (i) when all the queries are processed together; these results are provided in Table 5.3, (ii) when the queries are provided sequentially to the system, i.e., one after another; these

results are provided in Table 5.4. Querying times are seen to be similar across the RepL methods; the slight variation among the querying time of these methods is observed because of the difference in embedding generation time. The nearest neighborhood search time remains the same for all of the RepL methods. Since the `SuperVec(X)` models required for `H-SuperVec(X)` are generated in parallel, the querying time for `H-SuperVec(X)` remains approximately the same as for `SuperVec(X)`.

The RepL approaches provide a speed improvement up to $\backsim 50X$ vis-a-vis BLAST. When compared to MMseqs2 (run on 16 threads), we note that the real time is in a range similar to RepL methods whereas the system+user time of MMseqs2 is $\backsim 13X$ higher than the RepL methods. Like MMseqs2, optimizing our code to run on multiple threads is expected to reduce querying time (real-time) vis-a-vis MMseqs2; it remains a work to be considered in future. On analysing the querying time results further, we note that querying time for RepL methods reported in Table 5.3 is dominated by the embedding generation time ($\backsim 14sec$) while the neighborhood search time is minimal ($\backsim 1sec$). The overall querying time of the RepL methods is thus can be further improved by optimizing the embedding generation code to run on multiple threads.

We also note that the retrieval of similar sequences from a database for a query varies considerably based on the size and composition of the query and database, especially for alignment-based methods. Therefore, for further comparison of alignment and alignment-based methods we provide the distribution of BLAST, MMseqs2, `BioVec` and `SuperVec(X)` in Fig 5.7. The distribution of `Seq2Vec` and `H-SuperVec(X)` is noted to be similar to the `SuperVec(X)`. As seen in Fig 5.7, the variance of `SuperVec(X)` is smallest among all; also, it performs better than other methods as the querying time for most of the queries in `SuperVec(X)` is limited to $50msec$. In contrast, for BLAST, querying time is higher, and many queries take between $1 - 2sec$. For MMseqs2 and `BioVec`, the distribution is almost similar, and most of the queries take less than 150 ms.

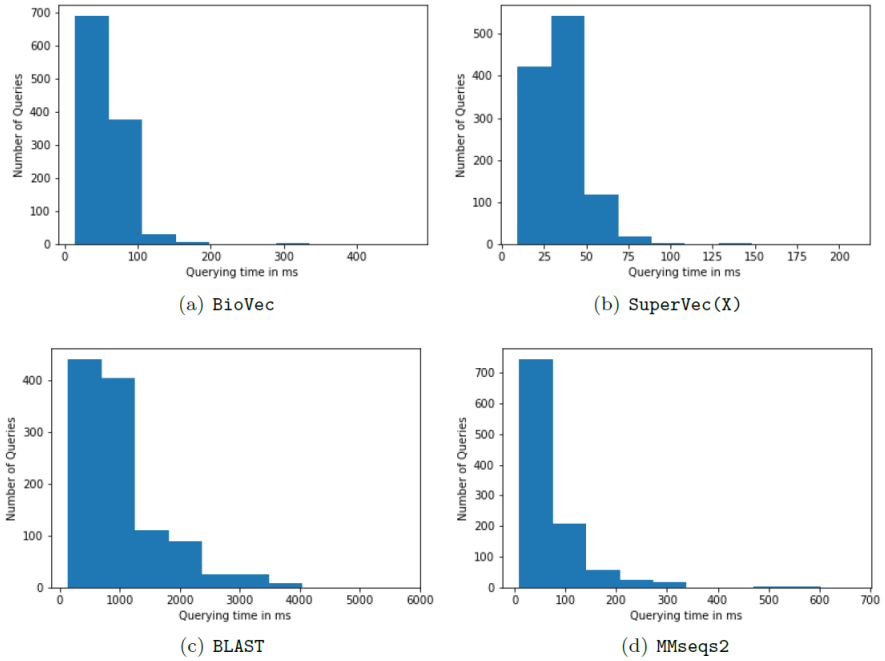
The advantages offered by `H-SuperVec(X)`—the gain in processing time and the improved retrieval results when compared to the earlier models (`SuperVec(X)`,

Table 5.3: **Querying Time*: Overall querying time for*** 1108 ***queries when processed together for the database of size*** 90*k* ***(200 classes) from dataset1.***

Methods	BioVec	Seq2Vec	SuperVec(X)	H-SuperVec(X)	BLAST	MMseqs2
real time	0*m*40*sec*	0*m*15*sec*	0*m*15*sec*	0*m*21*sec*	13*m*43*sec*	0*m*23*sec*
sys+user time	0*m*54*sec*	0*m*36*sec*	0*m*36*sec*	0*m*50*sec*	13*m*43*sec*	4*m*55*sec*

Table 5.4: **Querying Time*: Overall querying time for*** 1108 ***queries when processed serially for the database of size*** 90*k* ***(200 classes) from dataset1.***

Methods	BioVec	Seq2Vec	SuperVec(X)	H-SuperVec(X)	BLAST	MMseqs2
real time	1m1*sec*	0*m*39*sec*	0*m*39*sec*	0*m*42*sec*	19*m*31*sec*	1*m*2*sec*



(a) BioVec (b) SuperVec(X)

(c) BLAST (d) MMseqs2

Figure 5.7: **Querying time histogram*: Querying time histogram for different methods.***

BioVec, and Seq2Vec) —suggest that it may be combined with the higher precision method like BLAST to offer significantly faster retrieval of sequences with only a modest reduction in the fidelity of the results. Before discussing the proposed hybrid approach in next section, we summarize our observations related to the performance of different approaches as below:

- SuperVec and SuperVecX consistently outperform the baseline biological sequence embedding approaches on the sequence retrieval task for a wide range of database sizes.

- RepL methods are substantially faster than BLAST and provide a reduction up to $\backsim 50X$ in querying time, albeit with some loss of precision for larger databases (refer Fig 5.9).
- When compared to the MMseqs2, RepL methods provide comparable querying time. Also, these methods provide better precision than MMseqs2 for higher recall levels (refer Fig 5.9).
- The presence of a large number of sequences and classes in the database leads to deterioration in the performance of all methods. Although the retrieval performance of supervised methods also deteriorates for larger databases it still consistently performs better than `Seq2Vec` and `BioVec`, most likely as a result of its ability to incorporate the class label information in the sequence embeddings.
- In general, `SuperVec` performs better than `SuperVecX` for small class databases, whereas `SuperVecX` outperforms `SuperVec` for larger database sizes. We further observe that `SuperVec` performance is similar for both datasets 1 and 2, whereas the performance of `SuperVecX` is seen to degrade for dataset2 as compared to dataset1.

We now consider this hybrid approach in more detail.

5.8 Hybrid Approach: H-SuperVec(X)+BLAST

The proposed hybrid approach—`H-SuperVec(X)+BLAST` (`H(X)+BLAST`) follows a two-step retrieval process. `H-SuperVec(X)` is utilised initially to prune the original database to produce a list of results potentially relevant for the given query. Here the selection is based on nearest neighbour search in the database embedding space. The list of possible relevant subjects (the reduced database) obtained via `H-SuperVec(X)` is then provided to BLAST for re-ranking in accord with the given query. Fig. 5.8 shows the block diagram for the `H(X)+BLAST` approach, where DB represents the original database, DB_r the reduced database set obtained from the first step and DB_o the final ranked list of similar sequences for the given query.

The size of the DB_r is fixed by choosing a sufficient number of nearest neighbours (NN) for a given query. For our experiment we keep NN = $15k$, thereby

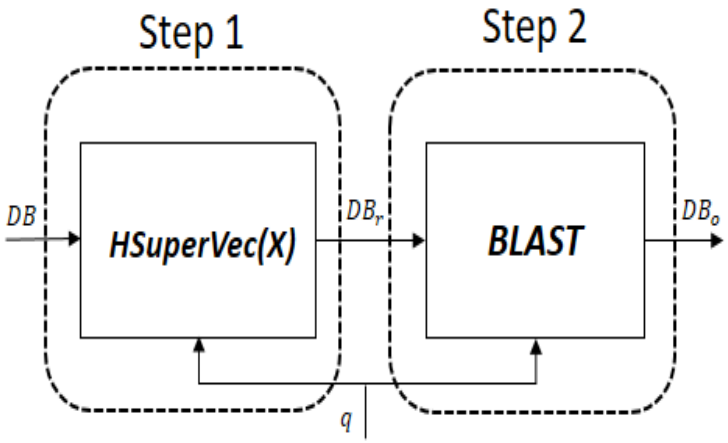
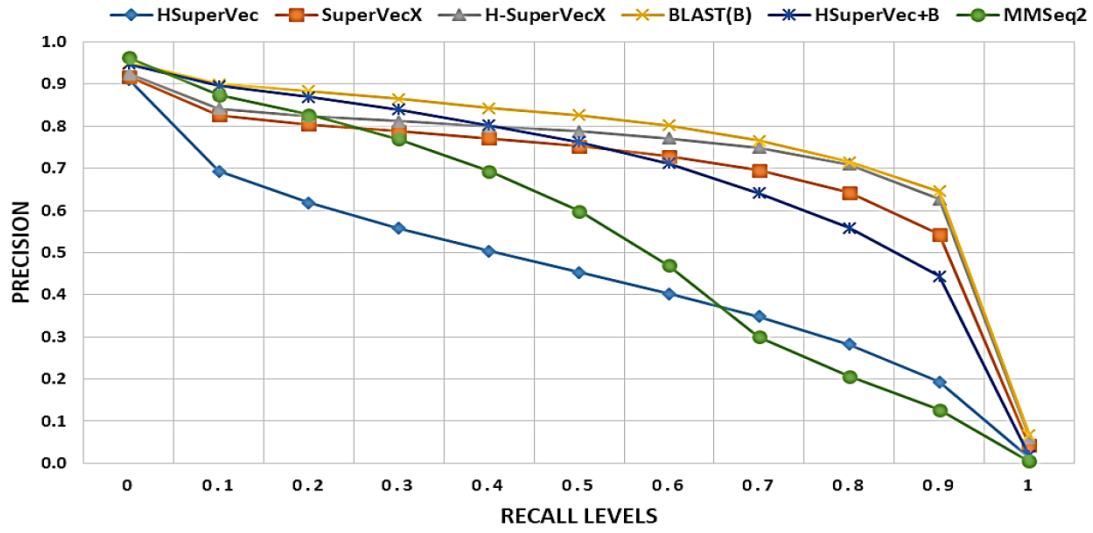


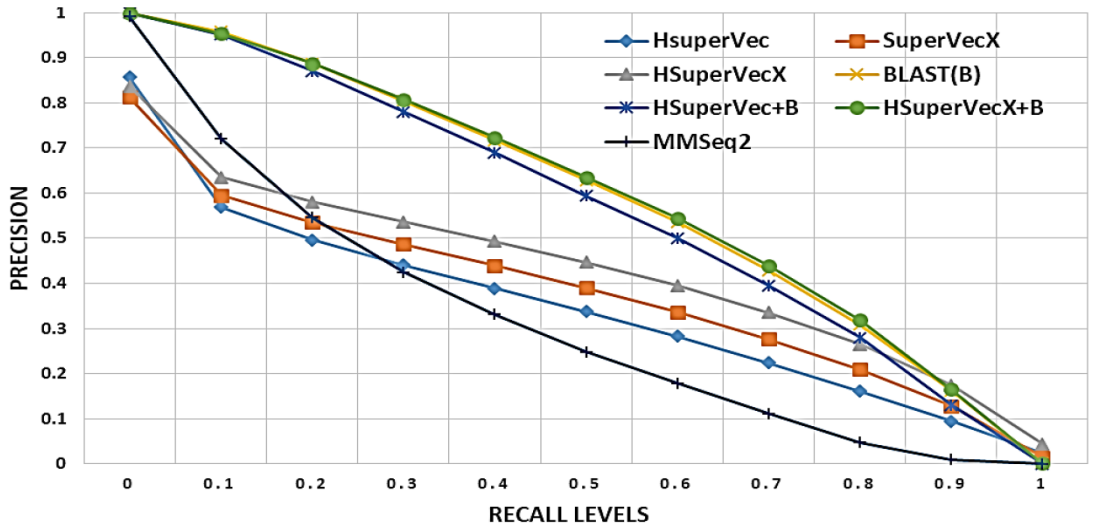
Figure 5.8: **Hybrid Approach** *: **Step1 uses** `H-SuperVec` **for pruning the original database (**DB**) and gives reduced database (**DB_r**). In step 2, BLAST re-ranks** DB_r ***based on alignment-based similarity between its sequences and the given query,** q ***and finally provide the list of retrieved sequences,** DB_o**.***

allowing BLAST to operate on a comparatively small database (DB_r). This provides a significant improvement in querying time compared to the direct application of BLAST to the original database. The average querying times using `HSuperVec+BLAST` and BLAST to process 1108 queries on a database of $\backsim$ 90k sequences are 947 *msec* and 1057 *msec* with a similar histogram (Fig 5.7). When repeated for $\backsim$ 60k queries on the same dataset, we obtained $\backsim$ 400 *msec* and 1 *sec* querying time for our hybrid approach and BLAST, respectively. Also, the use of BLAST in step 2 gives a performance improvement over `H-SuperVec(X)` on the retrieval task. With the increase in database sizes, we expect to see further improvements in querying time and retrieval performance of `H(X)+BLAST` as compared to `BLAST` and `H-SuperVec(X)` respectively. Utilising both approaches together thus gives us the best of both worlds —faster processing of queries and higher precision levels in the results.

Fig 5.9 provides a comparison of the hybrid approach with other embedding approaches, BLAST and MMseqs2 on the retrieval task for databases from dataset1 and dataset2. As shown, a substantial improvement can be achieved with the hybrid approach over the other embedding based approaches and MMseqs2, with the results obtained comparable in precision to BLAST.



(a) Dataset 1



(b) Dataset 2

Figure 5.9: **Retrieval result for hybrid approaches***: In (a) The results are averaged over* $\sim$ 60*k queries on the database of* $\sim$ 90*k sequences and 200 classes. The AUPR values for the methods shown are as follows, HSuperVec:* **0.451***, SuperVecX:* **0.703***, HSuperVecX:* **0.742***, BLAST:* **0.776***, HSuVec+B:* **0.701***, MMseqs2:* **0.535***. In (b) the results are averaged over* $\sim$ 38*k queries on the database of* $\sim$ 58*k sequences and* 100 *classes. The AUPR values for the methods shown are as follows, HSuperVec:* **0.343***, SuperVecX:* **0.381***, HSuperVecX:* **0.43***, BLAST:* **0.593***, HSuperVec+B:* **0.569***, H-SuVecX+B:* **0.597** *MMseqs2:* **0.311***.*

For a much larger database of $\sim$ 650,000 sequences (1886 classes) the results

for BLAST and HSuperVecX+BLAST are provided in Fig 5.10. Since the querying time required by BLAST is large ($\sim$ 1.42 secs per query), we show the result for 800 queries randomly chosen from largest two classes.

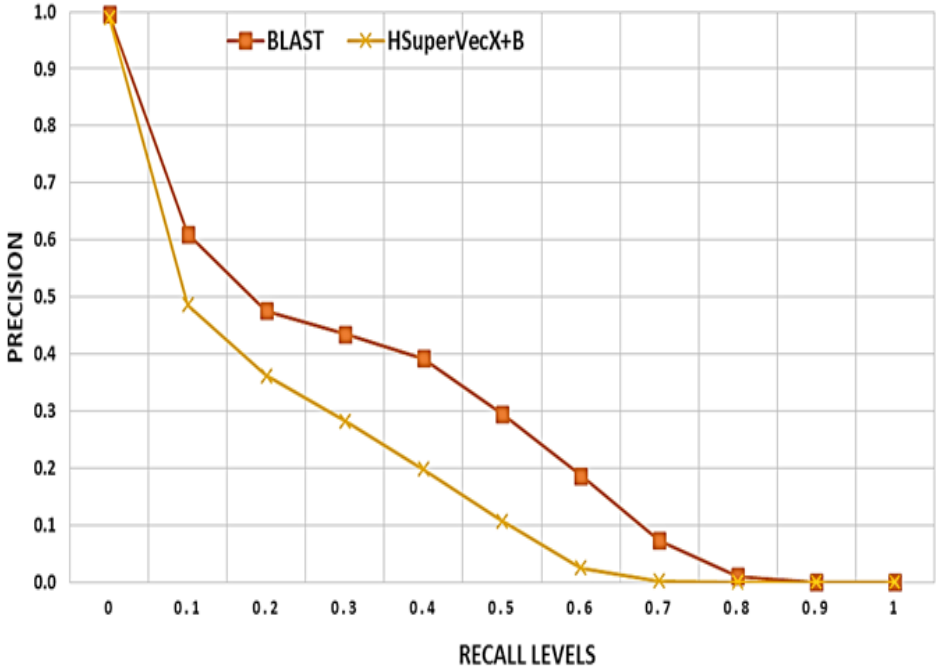


Figure 5.10: ***Retrieval performance comparison of hybrid approach and BLAST for database of size*** $\sim 650,000$ ***sequences (1886) classes from dataset1. The resuls are averaged over*** 800 ***queries randomly chosen from largest two classes.***

As shown, we can overcome the gap between HSuperVecX and BLAST by using the hybrid approach even for this large database. The average querying time for this database comes out to be 0.36 sec for HSuperVecX+B and 1.42 sec for BLAST, thereby giving a speedup of $4.5X$ in querying time. Note that, for the hybrid approach we chose the size of the reduced database to be $15k$. Increasing this size would further improve the performance at the cost of some increase in querying time.

Hybrid approaches are also expected to improve the sensitivity of RepL based methods. Here the sensitivity is defined in terms of the ability of a method to retrieve the maximum number of relevant sequences (true-positive) from the database before a non-relevant (false-positive) is extracted. This sensitivity is termed *AUC1*, which is defined as the fraction of true positive sequences retrieved prior to the appearance of the first false positive [30]. A comparison of

the cumulative distribution of *AUC1* values for different methods is provided in Fig 5.11, here the higher curve signifies higher sensitivity of the method. As shown, the hybrid approach (HSuperVecX+B) significantly improves the sensitivity of HSuperVecX and moves the curve close to BLAST.

Furthermore, there is some scope for further improvement in methodologies. Improved RepL approaches may be expected to yield better database and query embeddings, so that the relevant targets for a given query may be confined to a close neighborhood in the database embedding space. This would be likely to lead to improved performance on the retrieval task. Improvements in RepL approaches would also allow us to choose a smaller number of candidate relevant subjects for a given query, thus providing further gains in processing time and performance for hybrid approaches.

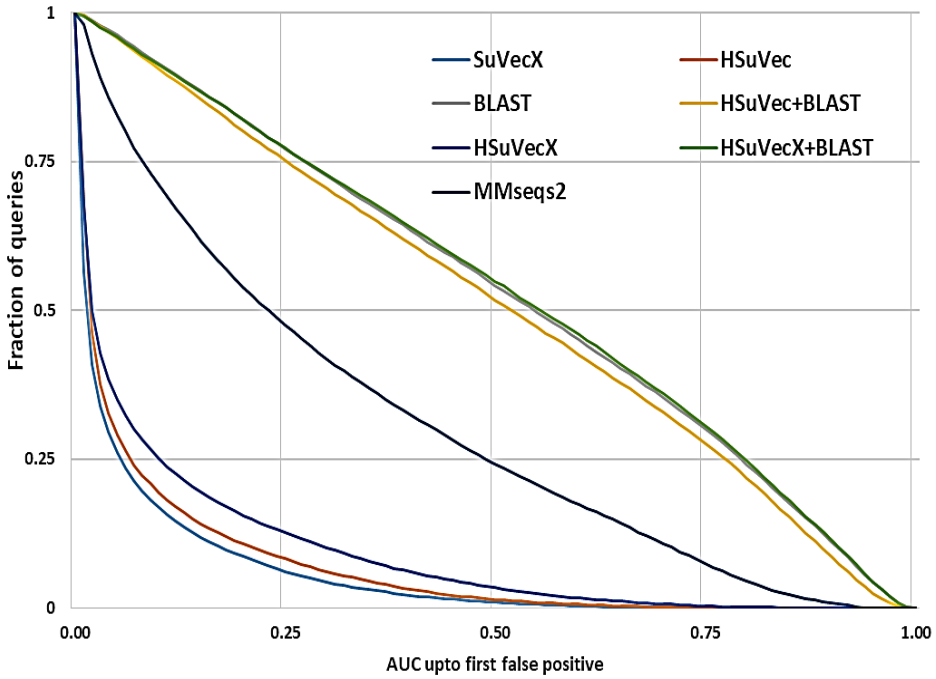


Figure 5.11: **Sequence searching sensitivity assessment** *: Cumulative distribution of area under the curve (AUC) sensitivity for* $\sim$ 38*k queries on the database of* $\sim$ 58*k sequences and* 100 *classes of dataset1. Higher curves signify higher sensitivity*

5.9 Conclusion

In this chapter, we demonstrated the utility of learned representations for the homologous sequence retrieval task. We showed that including meta-information improves the sequence embeddings and these are better suited for retrieval task compared to unsupervised embeddings. Not only are the proposed embedding models computationally inexpensive to train, but they also quickly generate embeddings for sequences. Lower computational requirements make such embedding based strategies suitable for the processing of large sequence databases. We showed that a combination of these models, as demonstrated by their hierarchical versions `H-SuperVec(X)`, further improves the performance for the database retrieval task. Given that the embedding models do not employ sequence alignment algorithms directly, there is a great scope for hybrid frameworks that use both alignment and non-alignment models to better represent and process biological sequences. Building on such hybrid approaches is a potential direction for further research.

Untill now, in this thesis, we have presented different methods for learning sequence embeddings through unsupervised (`Seq2Vec`) and supervised (`SuperVec(X)`) models. We demonstrated the utility of these embeddings for various classification (multi-class or binary) and retrieval tasks. These tasks can be categorized as single-entity inference problems, where the inference is made for single-proteins. Unlike single-entity problems, in bioinformatics, some problems deal with paired-entities. In such problems, mostly the task is to predict the relationship between two entities. Some of the examples of paired-entity problems are protein-protein interaction prediction, protein-DNA interaction prediction and drug-target interaction prediction. In the next chapters, we focus on one of these problems - the protein-protein interaction prediction problem and explore the potential use of embedding approaches in addressing this task.

6 Protein-Protein Interaction Prediction

Overview

In previous chapters, we mainly focused on proposing new representation learning techniques for biological sequences and demonstrating their usage for single-entity inference problems like protein family prediction and sequence retrieval task. We believe that representation learning approaches can also prove useful for paired-entity problems like Protein-Protein or Protein-DNA interaction prediction problems. This chapter concerns with one of these problems - the Protein-Protein Interaction (PPI) prediction task. Here we provide the preliminaries about protein interaction and methods to determine them. We cover the details for both experimental and computational PPI prediction methods, along with their limitations.

Since in this thesis we limit our exploration to sequence and Gene-Ontology term based PPI prediction approaches (covered in chapter 7 and 8), we first review the earlier works related to sequence based methods. After that, to assist in the understanding of our work on learning representations for Gene Ontology terms in Chapter 8, we provide details of the Gene Ontology project. Finally, we also discuss the issues related to the evaluation of the methods proposed for PPI prediction.

The organisation of this chapter is as follows.

In section 6.1, we discuss significance and challenges in predicting protein-protein interactions. The subsections 6.1.1 and 6.1.2, provide details of experimental and computational approaches respectively. In the section 6.2, we discuss the past works related to the sequence based approaches. After that,

in section 6.3, the details of GO term based prediction methods is covered, this section also include the details of gene ontology. In section 6.4, we discuss about evaluation of PPI prediction methods. Finally, we conclude the chapter in section 6.5.

6.1 Determining Protein-Protein Interactions

Proteins play a central role in cellular function, often mediated through the protein interactions leading to the formation of transient or stable complexes. Protein interactions may be ”functional” (indirect) or direct. Proteins that are part of a complex (a group of proteins) but do not directly interact are said to be involved in functional interaction. The direct interactions, on the other hand, are physical binary interactions and can be defined as *“specific physical contacts between protein pairs that occur by selective molecular docking in a particular biological context”* [100]. Protein-Protein Interactions (PPIs) vary in their duration and character, being governed principally by electrostatic forces and constrained by molecular structure [100]. Knowledge of PPIs can help determine the function of new proteins and improve our understanding of cellular processes such as DNA transcription and metabolic cycles [101].

Predicting PPIs is a challenging problem, and a number of different approaches have been proposed. These methods are broadly categorized as (i) experimental or (ii) computational methods. In the sections below, we discuss both of these approaches in turn.

6.1.1 Experimental Methods

Up to now, various experimental techniques have been developed for identifying and characterizing protein interactions. These methods either (i) detect direct or physical interactions between proteins, or (ii) measure interactions among a group of proteins - also known as “Co-complex” methods [102] (Fig. 6.1). Co-complex methods may be used to determine both direct and indirect interactions. It is challenging to infer the direct interactions from the output of Co-complex methods and algorithms like Spoke [103] are employed for this purpose. Some of the prominent high-throughput methods enabling

us to screen a number of proteins in a cell include yeast two-hybrid (Y2H) and tandem affinity purification coupled with mass spectrometry (TAP-MS).

Experimental methods for determining PPIs can further be differentiated based on (i) their ability to process small or large scale data, (ii) whether the processing is done in vitro or in vivo (iii) type of interaction (direct or indirect) and, (iv) type of characterization. Summary and categorization of various experimental methods are given in Table 6.1.

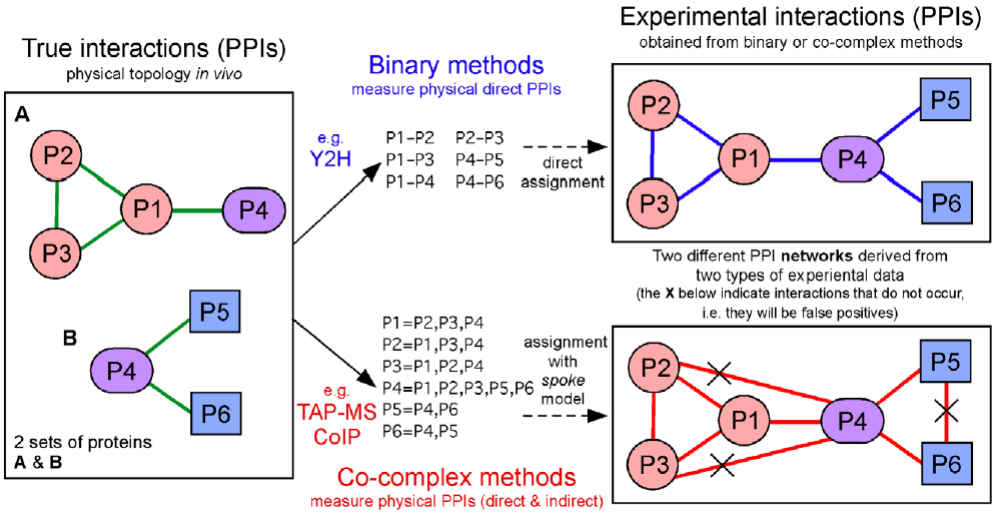


Figure 6.1: ***Approaches to determine PPIs - Binary and Co-Complex Methods. This figure is reproduced from [100].***

A detailed description of these methods is provided in [100]. Although high-throughput methods have provided a large amount of interaction data for analysis and verification, our understanding of interactions of many organisms remains incomplete. Also, the experimental techniques are prone to biases toward the localization and type of particular proteins. Other than these issues, the other significant drawbacks of these methods are: (i) such methods are expensive and time-consuming, and (ii) they are known to exhibit high false-positive rates [119]. However, it is important to note that none of the computational methods would be possible without the experimental work that provided the data in the first place.

A brief description of computational approaches is provided in the section below.

Table 6.1: ***Positives (+) in second columns indicate if the method is high-throughput and negative (-) otherwise. Second column indicates if the method can provide interaction information in vivo or in vitro. Type of interactions - physical interactions in a complex ("complex") or only pairwise interaction ("binary") is captured in fourth column. Inference of physical interactions through functional association is also indicated. Finally, in last column, type of protein characterization corresponding to each method is given. This table is reproduced from [104].***

Method	High-Throughput Approach	Living Cell Assay	Type of Interaction	Type of Characterization
Y2H [105]	+	In vivo	Physical (binary)	identification
Affinity purification - MS	+	In vitro	Physical (complex)	identification
DNA microarrays/Gene coexpression	+	In vitro	Functional association	identification
Protein microarrays	+	In vitro	Physical (complex)	identification
Synthetic lethality	+	In vivo	Functional association	identification
Phase display	+	In vitro	Physical (complex)	identification
X-ray crystallography	-	In vitro	Physical	structural and biological characterization
NMR-Spectrometry	-	In vitro	Physical	structural and biological characterization
Fluorescence resonance energy transfer	-	In vivo	Physical (binary)	Biological characterization
Surface plasmon resonance	-	In vitro	Physical (complex)	kinetic, dynamic characterization
Atomic force microspcopy	-	In vitro	Physical (binary)	Mechanical, dynamic characterization
Electron microscopy	-	In vitro	Physical (complex)	structural and biological characterization

6.1.2 Computational Approaches

Computational methods were largely motivated by the limitations of experimental methods. The earliest computational approaches were based on utilizing varied information sources related to proteins such as the protein domains [106, 107, 108], gene-co-expression data and tertiary structures [109, 110, 111]. Some other methods also explored the use of functional annotations (GO labels) [112, 113] for predicting PPIs. Although these computational methods demonstrate promising results, they can be applied to only those protein pairs for which the desired meta-information or annotation is available. Such meta-information may not be available for every protein considered. In contrast, with the advances in sequencing technology, the amino acid sequence (primary structure) of the proteins is readily available, making them a preferable input for PPI prediction approaches. These methods are collectively called as *sequence-based* approaches for PPI prediction.

One of the most prevelant ways to set a PPI prediction problem is to consider it to be a binary classification problem considering known PPIs as the "positive class". The challenge in setting up PPI problem as a supervised machine learning task is the unavailability of non-interacting (negative class) samples. A number of approaches have been proposed in the past to construct

negative-class examples, mainly based on the known localization of protein samples and random selection of protein-pairs that have not been identified as interacting pairs. Localization-based construction of negative class is argued not to generalize, as it does not consider the protein pairs that share the location but do not interact. The underlying idea behind the construction of negative class examples by random selection is based on the fact that only one among 300-1500 (depending on organism) random protein pairs, is estimated to interact [114].

But, even such negative class construction is not free from the possibility of choosing actual "positive" samples in the "negative class", which in turn would affect the training and produce results on test data which may not be completely accurate. A Venn diagram to understand the complete picture of PPI prediction as a binary classification problem is given in Fig. 6.2.

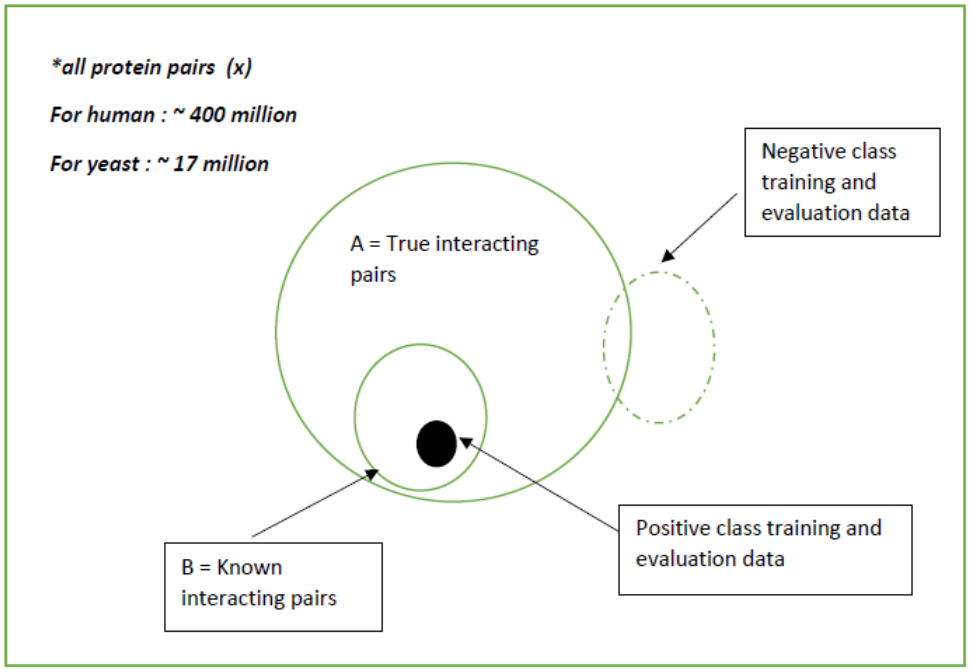


Figure 6.2: ***Venn diagram showing the overlap of negative labelled data in training and test sets with interacting and non-interacting protein pairs. x is set of all protein pairs in an organism. A is set of true interacting pairs of proteins. $x \setminus A$ is the real set of non-interacting pairs. To construct a negative set, pairs are randomly chosen from $x \setminus A$. Since the -ve pair space is much larger than the +ve pair space (generally for one interacting pair there are 300-1500 non-interacting pairs), random selection gives mostly non-interacting pairs but overlap with A is expected. [114]***

PPI prediction is a challenging and important problem, and significant effort has been put to address different aspects of the problem such as:

- Finding best ways to construct the negative class.
- Providing different feature construction methods to represent protein pairs that can be used with ML algorithm.
- Developing approaches that are computationally efficient for providing interactome level predictions.

In this thesis, we focus on providing low dimensional feature vectors for protein pairs using sequences and Gene Ontology (GO) terms for PPI prediction. These approaches have obvious computational advantages and may provide improved results. In the sections below, we review various sequence based PPI prediction methods that primarily rely on new ways of constructing a feature vector for protein pairs from their corresponding sequences. Subsequently we discuss GO term based PPI prediction methods.

6.2 Sequence Based Approaches

Most sequence-based approaches rely on the construction of n-dimensional feature vector to represent the protein sequence. This representation then can be used as an input to machine learning algorithm. Such feature vectors essentially summarize specific attributes of a protein sequence – such as the distribution of the amino acids, their physico-chemical properties [115], or the composition of a localized region [116, 117]. To apply machine learning approaches for the PPI prediction task, the protein pair needs to be represented as single mathematical entity. Generally, such a representation for a protein pair is obtained by concatenating the feature vectors of its constituent protein sequences, hence keeping the information derived from them intact.

Most prominent feature extraction methods include Auto Covariance (AC), Auto Cross Covariance (ACC), Conjoint Triad (CT), and Local Protein Descriptors (LD). AC and ACC based feature vectors are derived by considering protein sequences as time series and computing their auto and cross-correlation properties. The length of the AC and ACC based feature vectors

is 420 and 2940, respectively. The conjoint triad method divides the physico-chemical properties of the system into seven categories based on their dipole and volume scale [118], allowing the representation of the protein sequence based on an array of category labels. Once converted, the feature vector of the sequence is given as the frequency of conjoint triads, i.e. the three category labels. For a protein pair, the CT method gives a 686-dimensional feature vector.

The Local Descriptor method uses the same categories of physico-chemical properties as CT. The feature vector is constructed from three descriptors - composition (C), Transition (T), and distribution (D) for local regions. The descriptors calculated for each of the local regions are then stacked together to give a 630-dimensional vector for each protein sequence. Other methods based on local regions include Multi-scale Local Feature descriptors (MLD) [119] and the Multi-scale Continuous and Discontinuous feature set (MCD)[120]. MLD uses the multi-scale decomposition of protein sequences to account for overlapping local regions, binary coding to produce a 1134-dimensional vector for a protein pair. MCD works on a similar principle as MLD, though it gives a 1764-dimensional vector.

Other methods may use techniques from signal and image processing to compute a feature vector for the sequence, among these approaches is the physico-chemical Property Response Matrix combined with a Local Phase Quantisation descriptor (PR-LPQ)[121] and Substitution Matrix Representation (SMR) [122]. These methods generally operate on an intermediate matrix representation of a sequence; PR-LPQ uses physico-chemical properties of amino acids, whereas SMR uses the BLOSUM62 [25] matrix to construct the representations.

On top of these feature vectors, a number of different machine learning approaches have been applied to characterise PPIs. These methods have included the Support Vector Machine [115], Random Forests [122] and autoencoders [123]. More recently, studies such as DPPI [124], DNN-PPI (Deep Neural Network framework for Protein-Protein Interaction Prediction) [125], and PIPR (Protein-Protein Interaction Prediction based on Siamese Residual RCNN) [126] have explored deep learning frameworks for PPI prediction.

Note that these newer deep learning-based approaches are end-to-end classification models and do not specifically focus on the feature construction technique. Also, given their deep architecture, these are complex and hence computationally expensive approaches. Since the success of PPI prediction methods is strongly dependent on the feature vectors extracted from the sequences, new methods for feature extraction remain an active area for researchers. Although the sequence encoding methods discussed above are commonly used by researchers for the PPI prediction problem, they suffer from three significant drawbacks:

- such methods require prior knowledge of physico-chemical properties.
- the properties used are not guaranteed to cover the PPI interaction information adequately.
- the resulting feature vectors are usually of high dimension, increasing training and inference time and potentially limiting the effectiveness of the model.

One potential solution to these problems is to learn sequence embeddings that capture biological information that may be present in the sequences themselves, and to use them to represent a protein pair. The earliest work on learning biological sequence embeddings by Asgari et.al [20] shows that it is possible to learn embeddings rich in biological information for sub-sequences (*k-mers*) directly using the sequence data. These authors and our subsequent work (discussed in chapter 3) demonstrate the utility of representation learning (RepL) techniques for the protein family classification task. Following these primary works, other studies have shown the effectiveness of RepL approaches for a range of other downstream bioinformatics tasks [72, 127, 128, 95] such as trans-membrane and localization prediction (also covered in chapter 4). Although learning of sequence embeddings and their usages for downstream bioinformatics tasks has been explored now for the last few years, to date such methods have not been studied in detail for PPI prediction problem.

We explore the utility of sequence embeddings for PPI prediction and compare them with the benchmark methods in chapter 7.

6.3 GO-term based PPI prediction

While sequences remain a popular choice for constructing feature vectors for protein pairs, few of these approaches also explore the construction of protein pair feature vectors using the functional annotations (like GO terms) associated with the constituent proteins. GO (Gene Ontology) [21] is a controlled vocabulary of biological terms, structured as a directed acyclic graph (DAG). Details are provided below.

6.3.1 Gene Ontology

Gene ontology (GO) is a formal representation of knowledge about gene products (proteins and RNA) and their attributes. The graph is an outcome of an ongoing effort by Gene Consortium [21] to standardize the vocabulary used for protein attributes in different research domains. GO consists of three independent ontologies, namely, the Molecular Function Ontology (MFO), Biological Process Ontology (BPO) and Cellular Component Ontology (CCO). MFO contains terms which describe the molecular activity of the protein, for example, the enzymatic activity. BPO includes terms that describe the biological processes (for example, binding regulation), in which a protein is involved. The terms in CCO describes the cellular localization of a protein, i.e. the cellular components where a protein might be active. All of these ontologies are arranged as a hierarchical tree-like structure and form a directed acyclic graph (DAG) also known as the GO graph or GO tree.

A DAG has two properties, (i) the links between nodes are directed, which means that the edges include the direction and hence the path between two nodes can be traversed in both directions and (ii) no node can be traversed twice while moving from a node to another. Nodes (GO terms) in these graphs describe attributes of the protein, whereas the links describe the relationship between these terms. For example, in Fig. 6.3, the links describe that the term at a lower level *is a* part of the attribute defined by the term connected to it at a higher level. The relationships in GO graphs are: *is a*, *part of*, *has part*, *regulates*, *positively regulates*, *negatively regulates*, and *occurs in*. Two essential properties of GO graphs to be noted here are :

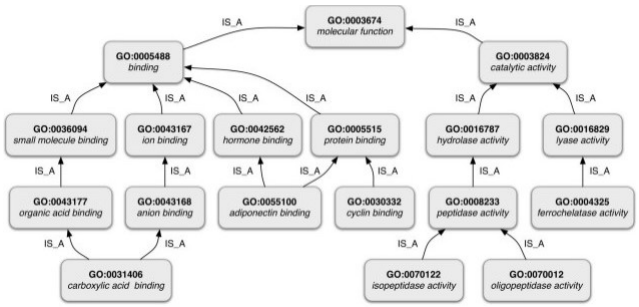


Figure 6.3: ***An Example from molecular ontology, nodes in the graph contains GO ids and their description (GO term), the links states the relationship between two nodes.***

- The root term explains the broad attribute, and the terms at lower levels describe more specific details.
- If a term is annotated to a protein, its parent terms also get automatically assigned to that protein.

As shown in Fig. 6.3, the molecular function term is the root; and as we traverse through the graph from root to leaf, the term describes more specific details. For example, "binding" (GO:0005488), is further categorized into different type of bindings, i.e., small molecules binding (GO:0036094), ion binding (GO:0043167), hormone binding (GO:0042562), protein binding (GO:0005515).

In past, studies [129, 130] have shown that there is a high chance that interacting protein pairs participate in similar biological processes, exhibit similar molecular functions or localized in the same cellular component. Most of the interacting proteins thus show a high GO semantic similarity.

Semantic similarity is often expressed as a distance between GO terms in the GO-DAG. These and alternative approaches are considered in [131], [132], [133], [134] and [135]. Many of these studies have shown that in most of the cases interacting proteins show high GO semantic similarity score. These approaches are unsupervised and rely only on Gene Ontology.

The supervised approaches on the other hand, construct a feature vector from the similarity score and use ML techniques for inferring PPIs. In one of the earliest approaches, Ben-Hur et al. [136], uses a kernel which combines the GO similarity score, BLAST score, and mutual clustering coefficient from PPI

network along with SVM to predict PPIs. Tastan et al. [137] used GO similarity values based feature vector for predicting viral host protein interaction.

Later, Maetschke et al. [112] proposed to use binary vectors to encode GO terms for representing protein pairs. These representations are binary vectors where all the indexes are set to zero except the index of the considered GO term. The length of these vectors is equal to the vocabulary size, where the vocabulary is the set of GO terms annotated to the whole proteome. To represent a protein pair, first, the set of terms corresponding to the individual proteins in a protein pair is derived. Once the GO terms for a protein pair is obtained, the binary encodings of these terms are summed to represent a protein pair. Maetschke et al. [112] tested different ways of deriving GO terms for a given protein pair from GO-DAG, the most straightforward being union of the GO terms associated with the individual proteins in a protein pair. Their other proposed methods include induction of GO terms from the GO-DAG along with the available annotations of individual proteins. The idea behind inducing GO terms is to capture the relationship between proteins. Bandopadhyay et al. [113] showed that these feature vectors could be further improved and hence they could also improve the prediction of PPIs. The feature construction approaches suggested weights the GO terms based on their position in the GO-DAG. One of the drawbacks of these representations is their size, which would not be suitable of large scale prediction of protein pairs. Considering this limitation, we propose to develop a framework to learn an enriched low dimensional representation of GO terms. We discuss in detail our work on learning GO term representation for predicting PPIs in chapter 8.

Most of the work on PPI prediction evaluates the prediction performance using a cross-validation strategy. Unlike single-entity problems, for paired input problems, like PPI prediction, the cross-validation results do not present the complete picture. A high cross-validation accuracy may not guarantee that the method would generalise across the population. Park et al. [138] discussed the issues related to evaluating performance for paired-input problems and proposed an alternative approach. In the section below, we provide the details of the strategy proposed by Park et al. [138] for evaluating PPI prediction methods.

6.4 PPI prediction Evaluation

In paired input problems, the inference is made for a pair of objects (i.e., the PPI prediction) rather than for the single object (e.g., protein family prediction). Based on the experimental results, Park et al. [138] showed that the performance of any method for the PPI prediction problem is affected by the presence of the individual proteins of the test sample in the training set. Better results are provided for the pairs that share protein(s) with the training data.

We cannot, therefore, expect a reliable assessment of generalisation if performance is evaluated on a test set dominated by samples which share components with the training data. Considering this issue, Park et al. [138] suggested that the paired-input prediction methods should be evaluated on three different test classes which they termed C1, C2, and C3. Here C1 constitutes the test samples for which both proteins are present in the training data, whereas, in C2, only those test samples are considered that share exactly one protein with the training data. Finally, the C3 test class is constructed from those test samples which share none of the constituent proteins with the training samples, making it the most challenging test class among the three. The process of spliting the datasets into training and C1, C2, C3 test classes is shown in Fig. 6.4

Park et al. [138] showed that the datasets previously used for cross-validation are close to the C1 type. They also noted that in the HIPPIE database [139], C1 type human protein pairs account for 19.2% of the samples, whereas C2 and C3 type samples account for much higher proportions - 49.2% and 31.6% respectively. Hence, in a typical cross-validation test set, the C1 class is more frequent than the corresponding population level, and performance estimates from cross-validation test sets cannot be expected to generalise reliably. Therefore, apart from the evaluation on the cross-validation and C1 test classes, results obtained on C2 and C3 test classes should be taken into account before making any judgment about the generalization properties of the method.

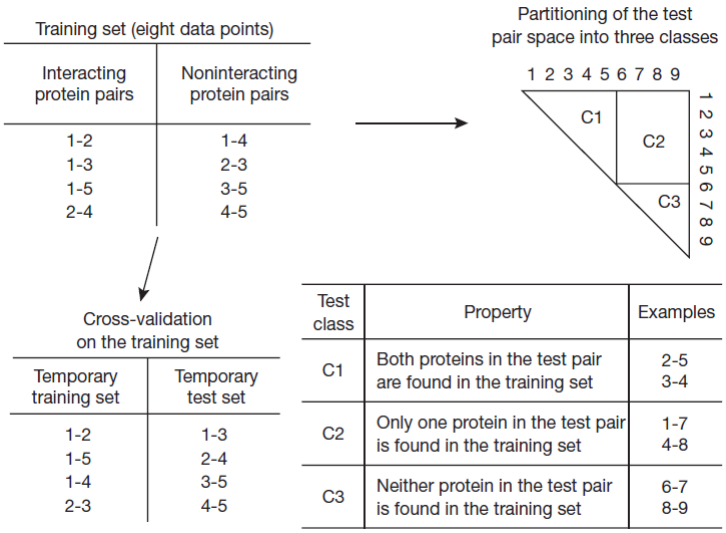


Figure 6.4: ***An Example (reproduced from [138]) to demonstrate how a dataset can be split into training and C1, C2 and C3 test sets***

6.5 Conclusion

This chapter provides a background for our works on PPI prediction tasks covered in the next two chapters. Here we provided the details of experimental and computational PPI prediction approaches and their associated advantages and limitations. Since we limit our focus on sequence-based (chapter 7) and GO term based (chapter 8) PPI prediction approaches, here we reviewed their literature in turn. We also discussed the suitable evaluation strategy for benchmarking PPI prediction methods, which we followed to report experimental results in upcoming chapters.

As noted in this chapter, most of the sequence-based PPI prediction work uses physico-chemical properties of the amino acids to obtain the representation of protein pairs. Such representations are large; also, the properties chosen for creating feature vectors might not be adequate to capture the interaction information. A possible alternative to such representations is to use the sequence embeddings for representing protein pairs. Considering this, in the next chapter, we explore the utility of embedding based representation of protein pairs for predicting protein-protein interactions.

7 Sequence representations and their utility for predicting protein-protein interactions

Overview

The success of machine-learning based methods for predicting PPIs heavily depends on the construction of the feature vectors used. Most of these methods use a set of physico-chemical properties derived from the sequence, whereas few work directly with the sequence itself. In the previous chapters, we showed that embedding a sequence in a low dimensional vector space has utility for tasks such as protein classification and sequence search. In this chapter, we explore these ideas in the context of PPI prediction task, making inferences from the pair instead of individual sequences.

We propose a generic PPI prediction framework that constitutes a representation learning module for feature construction and a binary classifier. To construct the feature vector for a protein pair, we concatenate the distributed representations (embeddings) learned for the sequences of the constituent proteins. Each protein pair is represented as a 200-dimensional feature vector. To learn the embedding of a sequence, we use `Seq2Vec` and `BioVec`. We also introduce a novel feature construction method which we call `SuperVecNW`. The embeddings generated through `SuperVecNW` capture network information to some degree, along with the contextual information present in the sequences. Finally, we feed these feature vectors into a Random forest classifier to predict protein pair interactions.

To show the efficacy of our proposed approach, we evaluate its performance

on *human* and *yeast* PPI datasets, benchmarking against the established methods. Furthermore, we test our approach on three well known networks: the one-core network (CD9), the multiple-core network (Ras-Raf-Mek-Erk-Elk-Srf pathway), and the cross-connection network (Wnt-related network) and demonstrate the improvement in predicting PPIs compared to the other methods.

Our experimental results show that naive low dimensional sequence embeddings provide better results on the PPI prediction task than most of the alternative representations based on other physico-chemical properties. These methods make fewer computational demands due to their lower dimensionality. Advanced representation learning methods that enrich the sequence embeddings with meta information are expected to improve the results further. The organisation of this chapter is as follows.

We begin by outlining the notation used in this chapter (section 7.1.1) followed by a brief description of PPI prediction problem (section 7.1.2). Following which, we cover details of our new representation learning method `SuperVecNW` in section 7.1.3.

Section 7.2 is devoted to the description of our proposed PPI prediction framework. Section 7.3 describes the experimental setup, which includes a description of the datasets and evaluation metrics used. Results of our proposed framework against benchmark methods on *human*, and *yeast* PPI datasets and three well-known networks are provided in section 7.4 and section 7.5 respectively.

Finally, we provide a general discussion around the PPI problem and possible approaches (section 7.5.1) and conclude the chapter (section 7.6).

7.1 Preliminaries

7.1.1 Notation

We denote M protein pairs as $\mathcal{P} = \{p_1, p_2, \ldots, p_M\}$, where p_z consists of two samples from a corpus of N protein sequences, $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$. We write $p_z = (s_i, s_j)$; to uniquely map the sequences with samples in $\mathcal{P}$, we add a

superscript to s_i i.e., p_z is written as $p_z = (s_i^z, s_j^z)$. We denote the vocabulary of L unique *k-mers* created from S as $\mathcal{K} = \{k_1, k_2, \ldots, k_L\}$ and each sequence $s_i = \left[k_{i_1}, k_{i_2}, \ldots, k_{i_{|s_i|}}\right]$ as an ordered list of *k-mers*, where $|s_i|$ denotes the count of *k-mers*. To avoid notation clutter, we use s_i also to denote its tag. Finally, the embeddings of *k-mer* k_i, sequence s_i and protein pair p_z are denoted as $\mathbf{k}_i$, $\mathbf{s}_i$ and $\mathbf{p}_z$ respectively.

7.1.2 PPI prediction problem

PPI prediction is a paired input problem in which a prediction is made about the relationship between two proteins. Since machine learning approaches generally operate on a single mathematical entity, for paired input problems, the pair is often represented as a combination of the feature vectors for the individual objects. In a setting where $\mathcal{P}_{train}, \mathcal{P}_{test} \subset \mathcal{P}$ are the training and test set of protein pairs respectively and $\mathcal{P}_{train} \cap \mathcal{P}_{test} = \emptyset$, the task in the PPI prediction problem is to determine whether or not the proteins in $p_y = (s_i, s_j) \in \mathcal{P}_{test}$ interact given that a label is available for the samples in the training set.

In our proposed framework, we use RepL methods, like `BioVec`, `Seq2Vec`, `SuperVecX` to generate the sequence embeddings that are concatenated for a protein pair, and a binary classifier for making predictions. The detailed descriptions of these embedding methods are provided in chapters 3 and 4. Before discussing the prediction framework in detail, we introduce a new representation learning method, `SuperVecNW`, that is based on our previously proposed method `SuperVec` (refer to section 4.2).

7.1.3 SuperVecNW

As explained before in section 4.2, the `SuperVec` model joins the two prediction units together to learn contextual and label information. The first unit follows the prediction of a *k-mer* given its context to capture contextual information. The second unit ensures the inclusion of label information in the training process by forcing the network to predict the tags of sequences that share the label with the considered *sample*. The joint cost function of `SuperVec` is given as below,

$$\mathrm{J}(\mathbf{S},\mathbf{K}) = \sum_{s_i \in \mathcal{S}} \sum_{j=1}^{n_i} \left(\underbrace{-\log\ \mathrm{P}r\left[k_{i_j} \mid C_{i_j},\ s_i\right]}_{NN1} - \gamma \sum_{z \in \mathcal{I}_i^+} \underbrace{\log\ \mathrm{P}r\left[s_z \mid s_i\right]}_{NN2} \right). \quad (7.1)$$

Note that this equation is earlier labelled as Eq. 4.1. Here $\mathcal{I}_i^+$ is the set of sequences that share the label with s_i. Note that NN1 and NN2 in Eq. 7.1 represent the two parts of the complete model, i.e., for learning contextual and label information, respectively. As noted before, `SuperVec` requires the labels of protein sequences for training purposes. In contrast to `SuperVec`, training of `SuperVecX` involves the prediction of the class label using all *k-mers* of the given sequences.

In the PPI prediction problem, along with the sequences, the information regarding relationships or their absence (interaction/non-interaction) between proteins in the training set is also available. Using a similar concept as utilised in `SuperVec`, we propose a new RepL model - `SuperVecNW` - that utilises contextual information and binary relationships of proteins to learn sequence representations. The description of `SuperVecNW` is given below.

`SuperVecNW`

In using `SuperVecNW` we make use of the available network information to label the sequences. The neighborhood of a sequence is looked up in the complete interaction network obtained from the training pairs of proteins. For a given sequence, the sequences connected to it in the PPI network are considered to come from the same class (interacting) and others from the different class (non-interacting).

For example, Fig. 7.1 shows a PPI network that is derived from given the training pairs. Here, s_3, s_1, s_7 and s_8 are from the same class (interacting w.r.t protein sequence s_3) and all others are the different (non-interacting) class. The joint cost function for `SuperVecNW` is same as for `SuperVec` as given in Eq. 7.1; the difference is only in the process of deriving class labels for a sample. For example in Fig. 7.1, for sample s_3, $\mathcal{I}_i^+ = s_1, s_7, s_8$ in Eq. 7.1.

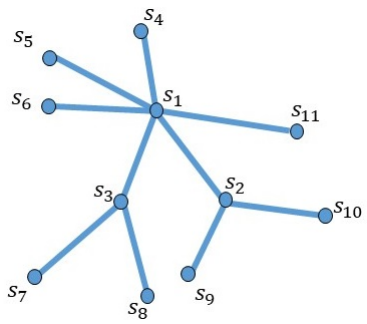


Figure 7.1: ***An example of a PPI network obtained from the training pair. Here*** s_i ***for*** $i = 1, 2, ...11$ ***represents the tag of the protein sequences. An edge between two proteins indicate that they interact, missing edge indicates the absence of known interactions.***

Table 7.1: **Context-word used per training sample (**s_i**,** k_{i_j}**,** C_{i_j}**) for different supervised representation learning models. NN denotes neural network.**

Models	NN	Architecture	context	word
`SuperVec`	NN1	CBOW	s_i and C_{i_j}	k_{i_j}
	NN2	Skip-Gram	$\{s_j, j = 1, 2..t\}$; s_i and s_j have same labels	s_i
`SuperVecNW`	NN1	CBOW	s_i and C_{i_j}	k_{i_j}
	NN2	Skip-Gram	$\{s_j, j = 1, 2..t\}$; s_i and s_j are interacting pair	s_i
`SuperVecX`	NN1	CBOW	$\{k_{i_j}; j = 1, 2....\|s_i\|\}; s_i \in l_i(label(s)\ for\ s_i)$	l_i

Note that all of these supervised models - `SuperVec`, `SuperVecX` and `SuperVecNW` - differ in their selection of *context* and *word* for the prediction task (refer Table 7.1).

7.2 Proposed Framework

In the proposed framework, along with the sequence embedding techniques discussed above, we use the Random Forest classifier for making predictions. Note that while the framework is generic and, in principle, any binary classifier can be used, we use the Random Forest (RF) Classifier [140] for our experiments due to its computational efficiency and convenience. An RF classifier requires only one hyper-parameter to be tuned and the method inherently avoids over fitting on the training data.

A Random Forest is a supervised classifier that uses an ensemble of decision

trees to make predictions. These decision trees are constructed using bagging (a bootstrap aggregation technique [140]), and random feature selection. In the bagging method, each of the decision trees is trained using randomly drawn subsets of the training set. For random feature selection, the nodes in each decision tree are split based on a random selection of m features drawn from the n features available. Here, $m <= n$. For a test sample, the prediction is made based on voting (averaging) the prediction output from each decision tree. The presence of multiple trees in the random forest helps to avoid overfitting and limits the error due to bias. Also, there is virtually no hyper-parameter to tune except for the number of trees.

In this work, we represent each sequence as a 100-dimensional feature vector, and the pair of proteins as a 200 dimensional vector. To use `SuperVecX`, we concatenate the sequences, with the result that the protein *pair* is represented by a *single* 100 dimensional feature vector. Note the much lower dimensionality of these representations compared to the other established feature vectors discussed in section 6.2. For interacting and non-interacting pairs, we use the class labels 1 and 0, respectively. Based on the results on a few training and test splits with different choices of the number of trees (100, 200, 500, 1000) – where each of the trees is built by selecting $m \ll 200$ random features at each node – we find 500 to be a good choice for the datasets used in this work. The prediction for a test sample is made based on the voting of the decision trees.

The complete pipeline of the proposed framework is provided in Fig. 7.2. Here block A shows the training stage of the representation learning model, A1; when trained, we denote it as A2. A1 is trained using the sequences in training set $\mathcal{S}_{train} \subset \mathcal{S}$. Note that $\mathcal{S}_{train}$ consists of all the unique sequences present in protein pairs in $\mathcal{P}_{train}$. Block B is used for making predictions. It is operated in two phases, namely, the training and testing phase. In the training phase, the samples (protein pairs) from $\mathcal{P}_{train}$ are passed through B1 to generate the embeddings of their constituent sequences. These embeddings are then concatenated to finally create the feature vector for the protein pairs, which are further used to train the binary classifier, B3. In the testing phase, similar to the training phase, the feature vector for any given test pair from $\mathcal{P}_{test}$ is generated by passing it through B1 and B2. This feature vector is

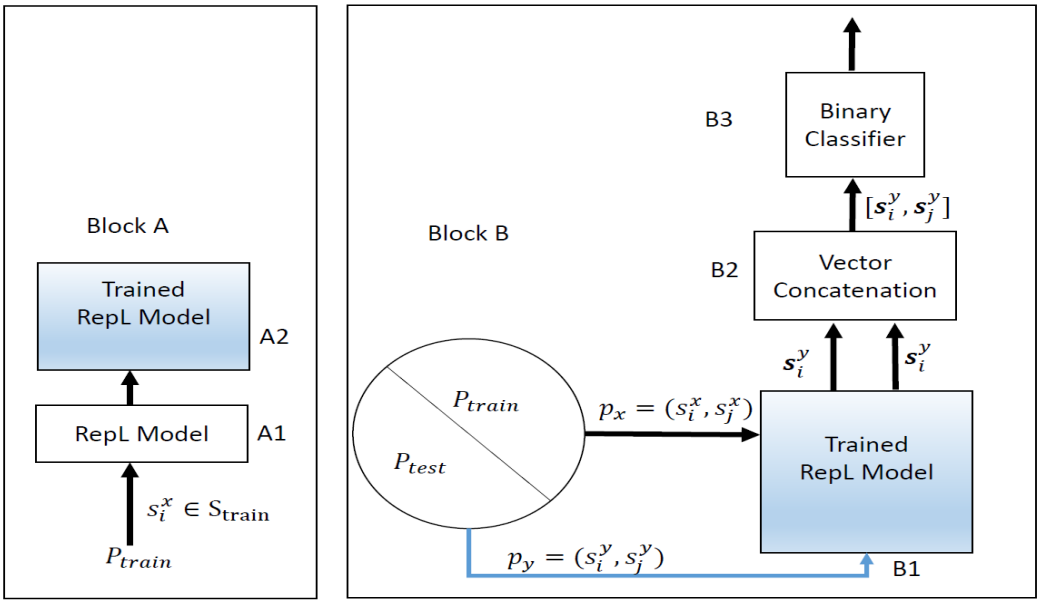


Figure 7.2: ***Proposed framework: Block A – training stage of representation learning model; Block B – constitutes a) Train/Test split of interacting/non interacting pair of protein sequences, b) vector concatenation module – for a pair of protein from Train/Test set, the vectors obtained from RL block are concatenated before finally passing to the c) binary classifier***

finally given to the trained binary classifier to predict whether the proteins interact.

In the section below, we describe the datasets and evaluation metrics used in this work.

7.3 Experimental Setup

7.3.1 Datasets

Dataset1: Park and Marcotte datasets

We first evaluate our method on the *human* and *yeast* PPI datasets provided by Yungki Park and Edward M. Marcotte [138]. These datasets were originally obtained from the protein interaction network analysis platform [141] and were further refined to contain only those sequences that share up to 40% identity. The *human* PPI dataset contains 20,117 proteins and 24,718 PPIs

whereas the *yeast* dataset contains 6,806 proteins and 14,938 PPIs. The experiments are conducted on the 40-fold train-test split for the CV, C1, C2, and C3 classes; for benchmarking, the split files were also as provided by [138].

Dataset2: Guo dataset

Guo et al. [115] provide PPI datasets for several species. These datasets were originally obtained from DIP [142] and then refined such that the majority of the remaining protein pairs share less than 40% pairwise sequence identity. These datasets are balanced and contain an equal number of positive and negative samples. For this work, we use the *yeast* dataset used as a benchmark in many of the earlier studies [124, 121]. In this dataset, there are 2450 unique proteins, and a total of 11188 protein pairs, where half of these pairs are interacting (positive pairs) and rest are non-interacting (negative pairs). For negative pairs, the proteins that have no evidence of interactions are randomly selected, and are further filtered based on their sub-cellular location.

7.3.2 Evaluation Measurements

To compare the performance of our approach on Dataset1 with other methods we use the performance measures Area under Receiver Operating Characteristic curve (AUROC) and the area under the Precision-Recall curve (AUPRC). These measures were used by Park et al. [138] to benchmark the performance of previous studies. The ROC is a graph showing the relationship between the False positive rate (FPR) and the True positive rate (TPR). In contrast, the PRC is the graph showing precision and recall values calculated at different thresholds. The AUROC and AUPRC summarize the ROC and PR curves, respectively. The AUROC value indicates the ability of the classifier to distinguish between two classes, whereas AUPRC shows the trade-off achieved between precision and recall. The AUROC and AURP are evaluated for all 40 (train-test) splits of Dataset1, and the mean value is reported along with standard deviation.

The benchmark methods that use Dataset2 for performance evaluation use: accuracy, precision, sensitivity, specificity, F-score, and Matthew's correlation

coefficient (MCC) as evaluation metrics. To compare with benchmark methods, we also evaluate our proposed approach for Dataset2 on these metrics. These are defined as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{7.2a}$$

$$Specificity = \frac{TN}{TN + FP} \tag{7.2b}$$

$$FPR = 1 - specificity \tag{7.2c}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.2d}$$

$$Precision = \frac{TP}{TP + FP} \tag{7.2e}$$

$$Fscore = 2 \times \frac{Precision \times Sensitivity}{Precision + Sensitivity} \tag{7.2f}$$

$$MCC = \frac{TP \times TN - FN \times FP}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}} \tag{7.2g}$$

Here the True Positive (TP) and True Negative (TN) counts record the number of correctly predicted interacting and noninteracting protein pairs. Similarly, the False Negative (FN) and False Positive (FP) counts reflect the number of incorrectly predicted non-interacting and interacting pairs, respectively.

7.4 PPI prediction Results

In this section, we present the results of our approach on the benchmark PPI datasets, *human* and *yeast* datasets (Dataset1) provided by Park et al. [138] and on *yeast* dataset (Dataset2) provided by Guo et al. [115]

7.4.1 PPI prediction for Dataset1

In these experiments, we compare the results of the proposed framework with the benchmark methods (M1-M8) that rely on physico-chemical properties for encoding the protein sequences. Here, M1 and M2 are signature-product based methods proposed by Martin et al. [143]. M1 uses the signature-product directly, whereas M2 [144] uses metric learning on the signatures to compute the feature vector for the protein pair. M3 [118] is the Conjoint Triad method;

in this method, the feature vector of a protein sequence is based on the frequency of conjoint triads. M4 and M5 both use the AC method proposed by Guo et al. [115] for feature construction, but they differ in the classification algorithm, with M4 using the SVM whereas M5 employs a random forest. The M7 method was developed initially for protein-RNA interaction prediction and is based on maximizing the scores computed for interacting protein pairs vis-a-vis non-interacting protein pairs within the vector space. M8, proposed by Ding et al. [145], calculates mutual information-based features based on the functional categories of amino acids. For predictions, M1-M4 use the SVM while M5-M8 use a random forest classifier. All of these methods (M1-M8), except M6 (PIPE), are machine learning-based approaches. PIPE is a sequence similarity-based approach that relies on the shared local similarity between known interacting protein pairs and the test pair. In the comparison for the *human* dataset, we also report prediction results for another sequence similarity-based approach, SPRINT [146]. In contrast to the machine learning-based approaches, these methods use only positive (interacting) pairs for making predictions and hence do not require the negative (non-interacting) examples.

Our approach is a generic machine learning based framework in which we can accommodate different representation learning methods for generating the feature vectors. We report the results for four different representation learning techniques – `Seq2Vec` (M9), `BioVec` (M10), `SuperVecNW` (M11) and `SuperVecX` (M12) – on both the *human* and *yeast* datasets.

Results

Table 7.2 and Table 7.3 summarize the performance of each method on the *human* and *yeast* datasets respectively. We report the mean AUROC and AUPRC and their corresponding standard deviations calculated over forty randomly selected train-test splits.

On the *human* dataset, for the CV and C1 test set, the performance of our methods (M9, M11, and M12) is better than M1-M7 except for M2. Our methods also give comparable performance to M8 and SPRINT. When compared to the conjoint triad (M3) method, we achieve an improvement of 18%.

Table 7.2: ***Comparison of the prediction performance between the proposed methods and other state of art methods on the* human *dataset from Dataset1***

Method	AUROC				AUPRC			
	CV	C1	C2	C3	CV	C1	C2	C3
M1 (Martin et al. [143])	0.81 ± 0.01	0.81 ± 0.01	0.61 ± 0.01	0.58 ± 0.03	0.82 ± 0.01	0.82 ± 0.01	0.60 ± 0.01	0.57 ± 0.03
M2 (Vert et al. [144])	0.85 ± 0.01	0.85 ± 0.01	0.60 ± 0.01	0.58 ± 0.02	0.85 ± 0.00	0.85 ± 0.01	0.60 ± 0.01	0.56 ± 0.03
M3 (Shen et al. [118])	0.63 ± 0.01	0.64 ± 0.01	0.55 ± 0.01	0.50 ± 0.00	0.67 ± 0.01	0.67 ± 0.01	0.57 ± 0.02	0.52 ± 0.05
M4 (Guo et al. [115])	0.77 ± 0.01	0.77 ± 0.01	0.57 ± 0.02	0.53 ± 0.02	0.77 ± 0.01	0.77 ± 0.01	0.56 ± 0.01	0.53 ± 0.02
M5 (Park et al. [138])	0.81 ± 0.01	0.81 ± 0.01	0.59 ± 0.01	0.54 ± 0.03	0.82 ± 0.01	0.82 ± 0.01	0.59 ± 0.01	0.54 ± 0.02
M6 (Pitre et al. [147])	0.76 ± 0.01	0.77 ± 0.01	0.64 ± 0.01	0.59 ± 0.02	0.79 ± 0.01	0.79 ± 0.01	0.67 ± 0.01	0.59 ± 0.02
M7 (Bellucci et al. [148])	0.56 ± 0.01	0.56 ± 0.01	0.53 ± 0.01	0.54 ± 0.02	0.56 ± 0.01	0.56 ± 0.01	0.53 ± 0.01	0.54 ± 0.02
M8 (Ding et al. [145])	0.82 ± 0.02	0.82 ± 0.01	0.60 ± 0.02	0.57 ± 0.02	0.83 ± 0.01	0.83 ± 0.01	0.60 ± 0.01	0.56 ± 0.02
M9 (Seq2Vec)	0.82 ± 0.01	0.82 ± 0.01	0.59 ± 0.01	0.55 ± 0.02	0.83 ± 0.01	0.83 ± 0.01	0.59 ± 0.01	0.55 ± 0.02
M10 (BioVec)	0.81 ± 0.02	0.81 ± 0.02	0.59 ± 0.01	0.56 ± 0.01	0.80 ± 0.01	0.81 ± 0.02	0.59 ± 0.01	0.56 ± 0.01
M11 (SuperVecNW)	0.82 ± 0.04	0.82 ± 0.01	0.58 ± 0.01	0.54 ± 0.02	0.84 ± 0.02	0.83 ± 0.01	0.59 ± 0.01	0.53 ± 0.01
M12 (SuperVecX)	0.80 ± 0.01	0.80 ± 0.01	0.55 ± 0.01	0.52 ± 0.02	0.8 ± 0.01	0.8 ± 0.01	0.56 ± 0.01	0.53 ± 0.01
SPRINT (Yiwei Li et al. [146])	NA	0.82	0.66	0.61	NA	0.86	0.70	0.63
PIPR	0.8 ± 0.01	0.8 ± 0.01	0.57 ± 0.01	0.52 ± 0.01	0.80 ± 0.01	0.80 ± 0.01	0.57 ± 0.01	0.51 ± 0.01

For C2, the averaged AUROC value of all machine learning-based methods ranges between $0.55 - 0.61$ whereas PIPE (M6) and SPRINT provide comparatively better results, i.e. 0.64 and 0.66 respectively. For C3, the values of averaged AUROC for all methods range from $0.55 - 0.61$. Note that for a classifier that randomly predicts interacting and non-interacting pairs, the AUROC is expected to be 0.5 and hence none of these methods can be considered to perform very well for C3. This leaves considerable scope for improvement in the feature construction methods. The AUPRC values obtained for all methods follow the same trend as AUROC values. Overall, SPRINT provides the best results for the C2 and C3 test class.

The results obtained for all methods on the *yeast* dataset are in general consistent with the results obtained for the *human* dataset, except for PIPE, which deteriorates for C2 and C3. SPRINT results are not available for the *yeast* dataset.

Table 7.3: ***Comparison of the prediction performance between the proposed methods and other state of art methods on the* yeast *dataset from Dataset1***

Method	AUROC				AUPRC			
	CV	C1	C2	C3	CV	C1	C2	C3
M1 (Martin et al. [143])	0.82 ± 0.01	0.82 ± 0.01	0.61 ± 0.02	0.58 ± 0.03	0.83 ± 0.02	0.83 ± 0.01	0.62 ± 0.02	0.57 ± 0.03
M2 (Vert et al. [144])	0.83 ± 0.01	0.84 ± 0.01	0.60 ± 0.02	0.59 ± 0.03	0.84 ± 0.02	0.84 ± 0.01	0.61 ± 0.02	0.58 ± 0.03
M3 (Shen et al. [118])	0.61 ± 0.01	0.61 ± 0.01	0.53 ± 0.01	0.50 ± 0.01	0.65 ± 0.02	0.65 ± 0.02	0.56 ± 0.03	0.53 ± 0.07
M4 (Guo et al. [115])	0.76 ± 0.02	0.76 ± 0.02	0.57 ± 0.02	0.54 ± 0.03	0.76 ± 0.02	0.76 ± 0.02	0.58 ± 0.02	0.54 ± 0.03
M5 (Park et al. [138])	0.80 ± 0.02	0.80 ± 0.01	0.58 ± 0.01	0.55 ± 0.02	0.78 ± 0.02	0.78 ± 0.01	0.57 ± 0.02	0.54 ± 0.02
M6 (Pitre et al. [147])	0.75 ± 0.02	0.75 ± 0.02	0.59 ± 0.04	0.52 ± 0.04	0.75 ± 0.02	0.76 ± 0.02	0.60 ± 0.05	0.47 ± 0.07
M7 (Bellucci et al. [148])	0.58 ± 0.02	0.58 ± 0.01	0.54 ± 0.02	0.52 ± 0.03	0.60 ± 0.02	0.60 ± 0.02	0.55 ± 0.025	0.53 ± 0.02
M8 (Ding et al. [145])	0.82 ± 0.02	0.82 ± 0.01	0.62 ± 0.02	0.61 ± 0.02	0.84 ± 0.01	0.84 ± 0.01	0.64 ± 0.02	0.62 ± 0.02
M9 (Seq2Vec)	0.81 ± 0.02	0.81 ± 0.01	0.59 ± 0.02	0.58 ± 0.02	0.83 ± 0.02	0.83 ± 0.01	0.61 ± 0.02	0.59 ± 0.02
M10 (BioVec)	0.81 ± 0.01	0.81 ± 0.01	0.61 ± 0.04	0.61 ± 0.02	0.82 ± 0.01	0.82 ± 0.01	0.62 ± 0.03	0.62 ± 0.02
M11 (SuperVecNW)	0.82 ± 0.01	0.8 ± 0.01	0.58 ± 0.01	0.54 ± 0.01	0.84 ± 0.02	0.82 ± 0.01	0.59 ± 0.01	0.53 ± 0.02
M12 (SuperVecX)	0.81 ± 0.01	0.81 ± 0.01	0.55 ± 0.01	0.54 ± 0.02	0.81 ± 0.01	0.56 ± 0.01	0.54 ± 0.01	0.54 ± 0.01
PIPR	0.83 ± 0.03	0.84 ± 0.01	0.6 ± 0.02	0.56 ± 0.03	0.83 ± 0.03	0.84 ± 0.02	0.6 ± 0.02	0.55 ± 0.02

Comparing the performance of different RepL techniques i.e. `Seq2Vec` (M9), `BioVec` (M10), `SuperVecNW` (M11) and `SuperVecX` (M12) on the *human* dataset, we observe that for CV and C1, M9, M11 and M12 provide slightly better result than M10, whereas for C2 and C3 they provide comparable values. For the *yeast* data set, M9 and M10 provide comparable results. This difference in performance when compared to the *human* dataset might be due to the larger number of *human* protein interactions available for training as compared to the *yeast* dataset.

In summary, our proposed framework (tested with different RepL approaches) outperforms most of the standard methods for CV and C1 test splits, while giving comparable results for C2 and C3 in most cases. In contrast to other methods, the representations for protein pairs used in the proposed framework are simple to construct and provide computational advantages in training and prediction time due to their lower dimensionality.

7.4.2 PPI prediction for Dataset2

In these experiments, we compare the results of our approach (tested with M9 and M10) on Dataset2 with baseline approaches SVM-AC [115], kNN-CTD [117], EELM-PCA [149], SVM-MCD [120], MLP [150], RF-LPQ [121], SAE [123], DNN-PPI [125], DPPI [124] and SRGRU, SCNN and PIPR from [126]. The results for the baseline approaches are taken from [126].

Results

As shown in Table 7.4, M9 and M10 outperform many approaches ((SVM-AC, kNN-CTD, EELM-PCA, SVM-MCD, SAE) including deep learning methods (DNN-PPI, SRGRU) and perform at par with DPPI and SCNN.The only method that outperforms our approaches is PIPR with an improvement of approx 2.5% in accuracy. Note that PIPR is a deep learning based end-to-end method whereas our methods comprise a simple feature extraction method with a binary classifier. Also, it is important to mention that these results are reported in the cross-validation setting and hence should not be considered

to generalize for the complete dataset. It is appropriate to evaluate these further on the C2 and C3 test classes. Considering this, we evaluate PIPR on Marcotte's benchmark dataset (Dataset 1) and procedure as discussed below.

Table 7.4: ***Evaluation of PPI prediction on the* yeast *dataset (Guo's) based on 5 fold cross validation. Mean and standard deviation is reported in the table***

Methods	Accuracy (%)	Precision (%)	Sensitivity (%)	Specificity (%)	F1-score (%)	MCC (%)
SVM-AC	87.35 ± 1.38	87.82 ± 4.84	87.30 ± 5.23	87.41 ± 6.33	87.34 ± 1.33	75.09 ± 2.51
kNN-CTD	86.15 ± 1.17	90.24 ± 1.34	81.03 ± 1.74	*NA*	85.39 ± 1.51	*NA*
EELM-PCA	86.99 ± 0.29	87.59 ± 0.32	86.15 ± 0.43	*NA*	86.86 ± 0.37	77.36 ± 0.44
SVM-MCD	91.36 ± 0.4	91.94 ± 0.69	90.67 ± 0.77	*NA*	91.3 ± 0.73	84.21 ± 0.66
MLP	94.43 ± 0.3	96.65 ± 0.59	92.06 ± 0.36	*NA*	94.3 ± 0.45	88.97 ± 0.62
RF-LPQ	93.92 ± 0.36	96.45 ± 0.45	91.10 ± 0.31	*NA*	93.7 ± 0.37	88.56 ± 0.63
SAE	67.17 ± 0.62	66.90 ± 1.42	68.06 ± 2.50	66.30 ± 2.27	67.44 ± 1.08	34.39 ± 1.25
DNN-PPI	76.61 ± 0.51	75.1 ± 0.66	79.63 ± 1.34	73.59 ± 1.28	77.29 ± 0.66	53.32 ± 1.05
DPPI	94.55	96.68	92.24	*NA*	94.41	*NA*
SRGRU	93.77 ± 0.84	94.60 ± 0.64	92.85 ± 1.58	94.69 ± 0.81	93.71 ± 0.85	87.56 ± 1.67
SCNN	95.03 ± 0.47	95.51 ± 0.77	94.51 ± 1.27	95.55 ± 0.77	95.00 ± 0.50	90.08 ± 0.93
PIPR	97.09 ± 0.24	97.00 ± 0.65	97.17 ± 0.44	97.00 ± 0.67	97.09 ± 0.23	94.17 ± 0.48
M9 (Seq2Vec)	94.24 ± 0.48	97.35 ± 0.55	90.97 ± 0.91	97.52 ± 0.53	94.05 ± 0.51	88.68 ± 0.93
M10 (BioVec)	94.18 ± 0.30	97.16 ± 0.67	91.03 ± 0.67	97.34 ± 0.65	93.99 ± 0.31	88.55 ± 0.61

7.4.3 Comparison with Deep Learning approach - PIPR

Some recent work has demonstrated the applicability of deep learning based approaches for the PPI prediction task. All of these approaches are primarily based on the Convolutional Neural Network and have a deep architecture. The most recent of these approaches, PIPR [126], provides an end-to-end deep GRU (gated recurrent unit) based architecture for PPI prediction. The results reported [126] show an improvement in classification results (refer Table 7.4) compared to other benchmark methods – including other deep learning based approaches.

These results are computed following a 5-fold cross validation test on Guo's *yeast* dataset (Dataset2). As discussed before, the cross-validation results may not reflect the ability of the method to generalize over the complete dataset. We thus evaluate the performance of PIPR on *Marcotte's* dataset for C1, C2, C3 test classes along with the cross validation setting. The results for *yeast* dataset show improvement (2-3%) for CV and C1; for C2 the results are comparable but for C3 the figure is some 2-4% below our approaches. For the *human* dataset, its performance is generally 2-4% below our methods. These results show that like the other techniques PIPR does not perform well for C2 and C3

test sets. For the *human* dataset, it is not the best among the machine learning based methods.

In the section below, we evaluate our method for predicting interaction in the known networks. After that, we discuss the limitations of sequence-based approaches and possible opportunities for further improvement.

7.5 PPI Network Prediction

A good PPI prediction method is expected to perform well in predicting the interactions of known networks. Previously, Shen et al. [118] and Ding et al. [145] demonstrated the utility of their approaches in predicting the interactions of three different types of networks, namely, one-core, multi-core, and crossover network. The one-core network is a simple network that consists of a core protein that is connected to the other proteins radially. In a multiple-core network, there are core proteins that interact with other proteins. The crossover network is a combination of one-core and multi-core networks.

Here, we also evaluate our approaches on such networks and use the CD9 (a tetraspanin protein) network for one-core, the Ras-Raf1-Mek1-Erk1-Elk1-Srf pathway for multi-core, and Wnt-related network for the crossover network types. The CD9 network contains 16 PPIs, the Ras-Raf1-Mek1-Erk1-Elk1-Srf pathway network has 189, and the Wnt-related network contains 96 PPIs. For experiments, we follow the same process as described in Fig. 7.2.

For training, we use human PPI data provided by Guo et al. [115] and ensure that the core protein (CD9, Ras, Raf1, Mek1, Erk1, Elk1, SRF) and proteins from the Wnt-related network are not present. We compare the results with established sequence-based approaches and show the superior performance of our approaches (tested on M9 and M10); the results are provided in Table 7.5. Among M9 and M10, M9 provides better results. The predictions over the CD9, the Ras-pathway, and the Wnt-related network obtained using M9 are demonstrated in Fig. 7.3, Fig. 7.4 and Fig. 7.5, respectively; these networks are drawn using Cytoscape software [151]. The blue and red edges in these graphs show true and false prediction, respectively.

Table 7.5: ***Comparison of the prediction performance of different methods on PPI networks. The entries in each column gives the count of correctly predicted interaction for one of the network. The total number of interactions in CD9, Ras pathway and Wnt related networks is 16, 189 and 96 respectively.***

Methods	Networks		
	CD9 (16)	Ras pathway (189)	Wnt-related (96)
Shen et al. [118]	13	161	73
Ding et al. [145]	14	174	91
M9 (Seq2Vec)	16	185	93
M10 (BioVec)	16	183	87

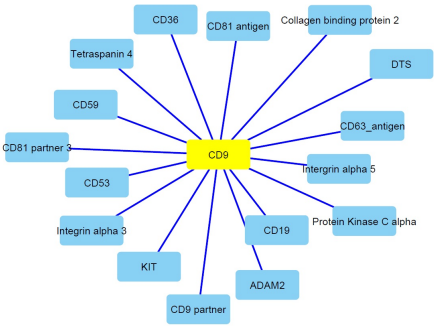


Figure 7.3: ***A one-core network for CD9. The blue edges show true predictions. The core protein is shown with yellow color, and its interacting partners are shown in light blue colour.***

7.5.1 Discussion

Predicting PPIs is a challenging problem. In recent years, computational methods have been a valuable alternative to experimental methods owing to their speed and relatively low false-positive rates. Over this period, sequence-based (machine learning) techniques have evolved, and better performance – as much as an 18% improvement over the earliest methods – has been achieved. Sequence-based methods mainly differ in the way the feature vector is constructed. Most of the available methods rely on known physico-chemical properties of the protein. Such properties might not cover the interaction information completely and hence may be limited in their value in distinguishing interacting from non-interacting pairs. The availability of sequence data and improvements in automatic extraction of features in text processing have motivated researchers to learn feature vectors directly from the sequences.

This work establishes that representation learning (RepL) based approaches

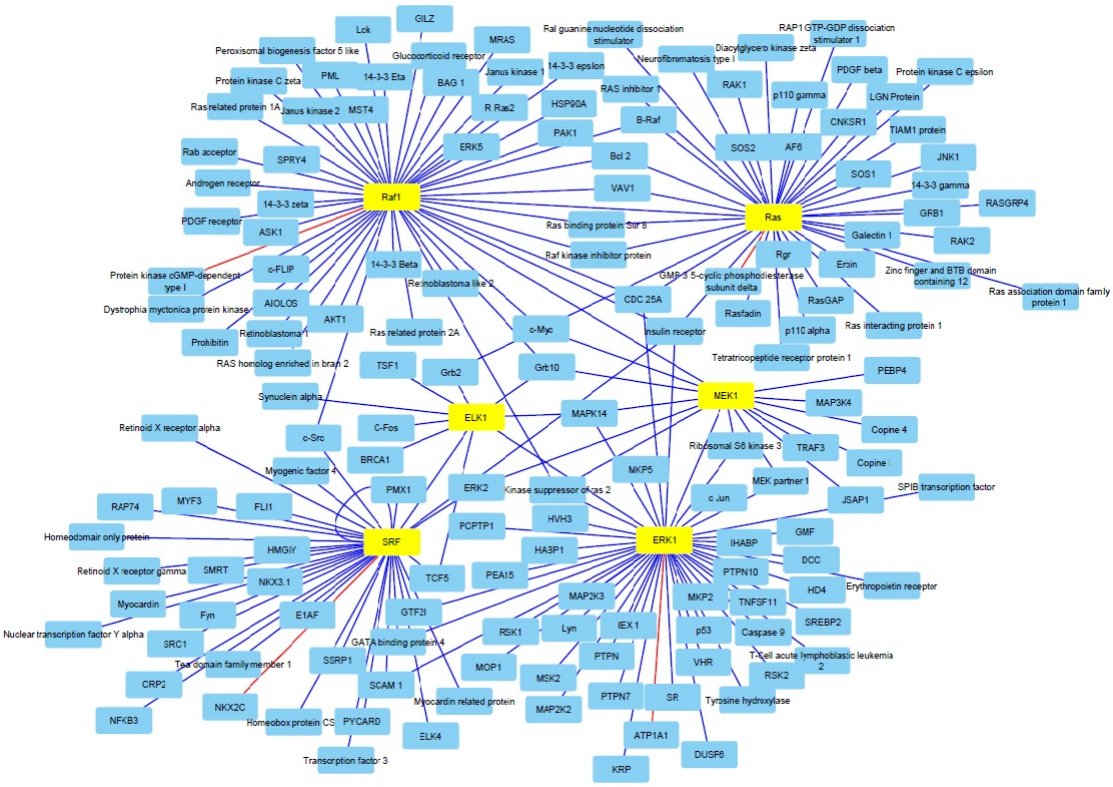


Figure 7.4: ***A multiple-core network for Ras-Raf1-Mek1-Erk1-Elk1-SRF. The blue and red edges show true and false predictions, respectively. The core proteins are shown with yellow colour, and its interacting partners are shown in light blue colour.***

can produce sequence embeddings that are useful for PPI prediction, in many cases providing better results than the physico-chemical property feature vectors without the need for extensive domain knowledge. Another advantage of using learned representations is their low dimensionality, which speeds up the training process and the inference step.

While sequence-based (machine learning) methods have evolved and have been shown to provide a high level of accuracy for PPI prediction, we need to be careful when making a judgment about their ability to generalize. Most of these methods give high cross-validation accuracy, which may not be a reflection on their generalization properties [138]. Therefore, we need to also focus on the results of the C2 and C3 test classes. In our experiments, we found that all methods – including the recent deep learning method PIPR – show good cross-validation accuracy, but when evaluated for the C2 and C3

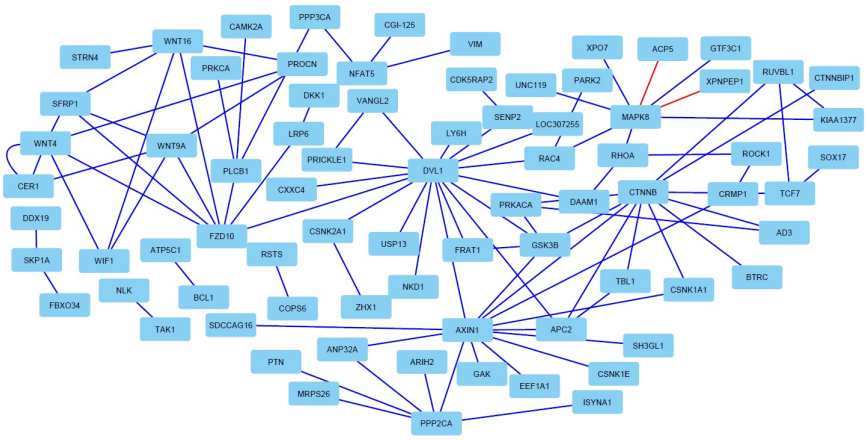


Figure 7.5: ***A crossover network for Wnt related pathway.The blue and red edges show true and false predictions, respectively.***

test class, their performance markedly deteriorates.

Some methods which rely directly on similarity – such as PIPE and SPRINT – show improved results for the C2 and C3 test class for the *human* dataset (refer Table 7.2), although there remains a considerable scope for improvement. For the *Yeast* dataset (refer Table 7.3), PIPE perform poorly on the C1, C2 and C3 test classes compared to other methods. Results for SPRINT on the *Yeast* datasets were not available. In contrast to machine learning approaches, PIPE and SPRINT work at a much more granular level, and use sub-sequence comparisons as the cornerstone for making predictions. These methods use the computationally intensive preprocessing step [146] of computing similar sub-sequences from the complete protein set. In contrast, machine learning approaches rely on feature vectors that capture the overall sequence information. Working at such a level of granularity proves to be advantageous for PIPE and SPRINT on the C2 and C3 test classes of the *human* dataset. Using embedding based approaches that work at different resolutions can potentially yield improvements for the C2 and C3 test cases.

All of the methods considered performed relatively poorly on the C3 test class. To understand this better (specifically for representation learning based methods) we undertook some exploratory investigations. These investigations were based on the interactions between proteins in the neighborhood of each protein in a potentially interacting pair. Here we focused on proteins from the CD9 network and made predictions for all possible protein pairs. There are

17 proteins in the CD9 network, which makes a total of 136 possible protein pairs. If we remove all of these 17 proteins from the training set, this set constitutes a C3 test class.

To label a protein pair as interacting or non-interacting, we use the well known biological database STRING [152], which scores the known protein-protein interactions by taking into consideration various evidence sources (experimental evidence, co-expression, mention in literature etc.). If a protein pair is present in the STRING database (> 0.4 score), we label it as interacting; otherwise it is labelled as non-interacting. The results we obtain with the embeddings in M9 and M10 show that these methods can detect most of the interactions but suffer from an high false-positive rate. We note that the protein pairs that are not present in the STRING database may be negative or include some interacting pairs which might not be documented. By examining the possible links between proteins in the neighborhood of each member of the pair, we sought to obtain some insight into the over-generalisation of many of these methods.

We selected candidates based on: (i) the distance of proteins from the test sample in the proteome and (ii) the interaction/non-interaction of the neighbours of the test sample proteins in the embedding space - constructed of the embeddings of unique proteins present in the training set. We found sufficient evidence to question 10-20% of the false positives. Still, the approach was inconclusive, and hence it requires further improvements to be considered as a potential strategy for improving representations. At present, the efficacy of the proposed approach is limited by the embeddings used to construct a feature vector for a protein pair.

Capturing the interaction information from a pair of proteins in a feature vector remains a challenging task. While our methods offer much more convenient construction of feature vectors than some existing alternatives, and performance remains comparable, some insight is lacking and there is room to improve our models for generating feature vectors from raw sequences.

One other possible opportunity lies in focusing on hybrid models that take advantage of available high-quality annotations and measurement data along with the sequence data. Such methods may be useful in dealing with the C2

and C3 test class samples. Also, we note that none of the sequence-based methods considers the PPI prediction in a specific context (e.g., different cell types). Having models that can utilise both measurement and sequence data can help us to make a better prediction of the PPIs in a specific context. Given that our supervised models (`SuperVecNW` and `SuperVecX`) for feature construction are flexible, it might be possible to extend these approaches for context-specific PPI predictions while retaining the low dimensional representation.

7.6 Conclusion

In this chapter, we proposed a framework that uses the embeddings learned through `Word2Vec` based models for the PPI prediction task. Such embeddings do not require any prior biological knowledge and are learned directly from the sequence data. Moreover, these embeddings offer a far lower dimensional representation than those based on physico-chemical properties of the proteins and their constituent amino acids.

The experimental results obtained on both *human* and *yeast* datasets, and on selected networks confirm that better sequence embeddings for PPI prediction tasks can be generated without necessarily relying on prior biological knowledge. We observe that most of the sequence-based methods perform well when evaluated on CV and C1 but not on the C2 and C3 test classes. The inability of sequence-based methods to perform better for C2 and C3 test classes presents an attractive and challenging opportunity to learn feature vectors that better represent the interaction between two proteins. Perhaps some limited use of prior knowledge in a supervised learning framework or working on a smaller resolution similar to SPRINT may provide the answer – maintaining most of the convenience of our approach while offering improved performance on the more challenging datasets.

Apart from the sequence data, the other critical data source that can be useful for PPI prediction is functional annotations (GO Terms) of proteins. To our advantage, over the past few years, the coverage of GO terms has also increased. One of the expected benefit of GO terms based approaches over sequence based approaches is better results for C2 and C3 test classes. On the downside, the coverage is still not 100%; therefore, for many proteins,

functional annotations are not available limiting the usage of GO-terms based method to the proteins that have annotations. Still, it is of value to explore the utility of GO terms for PPI prediction, which may further lead to the development of hybrid approaches (sequence+GO terms) for PPI predictions. In the next chapter, we explore different feature construction techniques for protein pairs that use GO for predicting PPIs. Also, we propose a new representation learning method for generating low dimensional embeddings for GO terms that provide improved results for predicting PPIs.

8 Gene Ontlogy terms and their utility for predicting PPIs

Overview

In the previous chapter, we noted that current sequence-based PPI prediction approaches do not perform very well on difficult test cases - C2 and C3. Their limited performance suggests that we should explore and benchmark the usage of alternative data sources to predict PPIs. One of the available data sources that can prove useful in making inference about protein interactions is the functional annotation of the proteins. Gene Ontology consortium provides a unified knowledge source for functional annotations of gene products; these annotations are usually known as GO terms. Over the years the coverage of Gene Ontology has also increased significantly.

Most of the available methods that utilise GO terms uses "bag-of-words" (BOW) based technique to represent a protein pair, which are then used as an input to an ML algorithm to make predictions. These representations are naive and hence allow some potential improvements. Also, such representations are of higher dimension (here the total number of GO terms) and therefore can pose computational challenges for large-scale prediction experiments. Considering these limitations of BOW based representations mentioned above, in this chapter, we explore the utility of learning low dimensional representations for GO term to predict PPIs. Here, we introduce a method for learning low dimensional embeddings for GO terms which are further combined to generate feature vectors for protein pairs. We use a binary classifier that is trained on the protein pair representations mentioned

earlier for making predictions. We compare both GO term and sequence-based methods on C1, C2 and C3 test cases in our experiments. The organisation of this chapter is as follows.

We begin in section 8.1 by discussing GO term based one-hot-encoding feature construction methods for representing protein pairs and notation used in this chapter. Next, in section 8.2, we introduce a method for learning low dimensional representations for GO terms. Successively, in section 8.3, we discuss the experimental setup and PPI prediction results for *human* and *yeast* datasets. Finally, in section 8.4, we conclude the chapter.

8.1 Representing protein pairs using GO labels

Constructing a feature vector for a protein pair using GO terms involves (i) selecting features (here GO terms) for a protein pair from the Gene Ontology and (ii) using them to generate the corresponding feature vector. The available methods have explored different approaches to select the appropriate GO terms for a protein pair encoding these through the usual bag-of-words (BOW) technique to construct the feature vector. These feature vectors are then used as an input to a machine learning algorithm for making predictions. In the sections below, we first provide the notation used in this chapter, followed by two bag-of-words based feature construction approaches.

8.1.1 Notation

We consider K GO terms present in the selected Gene Ontology: cellular localisation, biological process or molecular function (discussed in section 6.3.1). Each GO term is denoted as g_i and their collection as G. The set of available protein pairs is denoted as $\mathcal{PP}$. Note that these protein pairs may have common proteins. The collection of unique proteins that makes $\mathcal{PP}$ is denoted as $\mathcal{P}$. The embeddings for GO terms, proteins, protein pairs and the matrices corresponding to their collections are denoted in bold. For example, the embedding for i^{th} GO term is denoted as $\mathbf{g}_i$, similarly the collection of all GO terms i.e., G is denoted as $\mathbf{G}$. A summary of the notation used and other important definitions is provided in Table 8.1

GO terms	$G = \{g_1, g_2 \dots g_K\}$
Protein ids	$\mathcal{P} = \{p_1, p_2, \dots, p_N\}$
Protein pairs	$\mathcal{PP} = \{pp_1, pp_2, \dots, pp_M\}$
GO terms for p_i	$S_i = \{g_i^1, g_i^2, \dots, g_i^n\}$; $S_i \subset G$
All terms for p_i and p_j	AL(S_i, S_j) = $S_i \cup S_j$
All common terms for p_i and p_j	AC(S_i, S_j) = $S_i \cap S_j$
Parents for p_i	$parents(p_i)$
Parents of terms in S_i	$parents(S_i) = \cup_{g \in S_i} parents(g)$
Immediate Children's for p_i	$chldrn(p_i)$
Ancestors of S_i	$A(S_i) = S_i \cup A(parents(S_i))$ $A(\emptyset) = \emptyset$
Depth of a GO term	$D(g) = max\{D(u) u \in parents(g)\} + 1$ $D(r) = 0$, r is root
All ancestors	$AA(S_i, S_j) = A(S_i) \cup A(S_j)$
All Common Ancestors	$AA(S_i, S_j) = A(S_i) \cup A(S_j)$
Only Lowest Common Ancestor can be more than one with same depth	$OLCA(S_i, S_j) = argmax\{D(g)\}$ $g \in ACA(S_i, S_j)$
Lowest Common Ancestors	$ACA(S_i, S_j) = S_i \cup S_j \cup OLCA(S_i, S_j)$
Upto Lowest Common Ancestors	$ULCA(S_i, S_j)$ $= \{g \mid g \in AA(S_i, S_j) \wedge D(g) \geq d_{lca}$ d_{lca} : depth of lowest common ancestors

Table 8.1: ***Notation***

8.1.2 BOW based feature construction approaches

The process adopted for constructing a GO term based BOW feature vector for protein pairs is as follows. In the first step, the GO terms for the protein pair are selected from a Gene Ontology. So far, different methods have been proposed to represent protein pairs with GO terms. A few of them are defined in Table 8.1. Once the GO terms for a protein pair are selected, one-hot-encoding of these terms may be used to generate the feature vector for the protein pair.

For *One-hot-encoding* of i^{th} GO term, g_i we produce a binary vector of length K (vocabulary size), where all indices are zero except the i^{th} index which is set to one. We evaluate the utility of GO term embedding based protein pair feature vectors by comparing them with the following BOW based protein pair feature vector.

- NR (Naive representation) :
  The vector representation for pp_i is obtained by adding the one-hot encodings of the GO labels in $S_i \cap S_j$.
- ULCA (UptoLCA) [112]
  The pp_i is represented as the sum of the one-hot encodings of the GO labels in $ULCA(S_i, S_j)$

One of the limitations of BOW based protein pair feature vectors is their high dimensionality, which becomes more important if we wish to make proteome level predictions. Also, these vectors capture low-level information, mainly based on the presence and absence of the individual GO term for a particular protein pair. In this work, we hypothesize that it is possible to generate better representations for the GO terms and hence for the protein pairs, that are smaller in size and also capture abstract information. In the experimental results, we show that our proposed representations for protein pairs yield competitive results and computational advantages.

In the section below, we describe our proposed approach for learning representations of GO terms that are further utilised for predicting interacting protein pairs.

8.2 Representation Learning for GO terms

Our proposed framework for learning representations of GO terms involves (i) extraction of features for the protein pairs, (ii) using those features for training a shallow neural network. The complete framework is shown in Fig. 8.1. Here, Block A shows the feature extraction stage for a protein pair (p_i, p_j). As shown, first the GO terms corresponding to p_i and p_j, i.e. S_i, S_j are obtained from the set of all GO terms G. The sets of GO terms S_i and S_j are then processed to get a combined set of GO terms, simplest being $S_i \cup S_j$, which are then fed into Block B. In Block B, the input is obtained by averaging the vectors corresponding to the GO terms in the combined GO terms set.

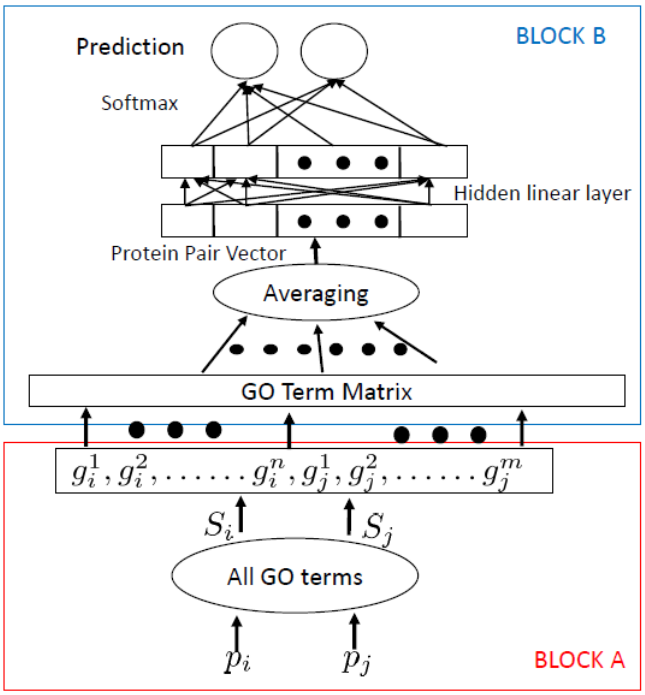


Figure 8.1: ***GO representation Learning: Block A – constitutes a) extracting GO terms corresponding to the individual proteins in the protein pair b) generating the GO terms set for the protein pair. Block – B is a feed-forward NN that is trained using gradient-descent approach. Once trained the low dimensional representations for GO terms is obtained.***

This averaged vector is then directly fed to the neural network for making interaction/non-interaction prediction. The NN is trained using the gradient descent algorithm; the vectors corresponding to the GO terms are updated in the process. The updated vectors obtained at the end of the training phase are

considered as embeddings of the GO terms. The embedding of protein pairs is obtained by combining the corresponding GO terms. A binary classifier is then trained using these embeddings to give a protein interaction predictor.

Once trained, the binary classifier is then directly used to make a prediction for new protein pairs. We consider both NR and ULCA feature extraction approaches for training the NN, calling them sNREmbd and ULCAEmbd respectively.

8.3 Experimental Setup and Results

We follow the same setup as described in the last chapter. Similar to the evaluation strategy adopted there, we evaluate and compare our proposed approach on C1, C2 and C3 test sets. For datasets, we use a *human* and *yeast* dataset obtained from [138]. To compare the performance of different methods, we use the performance measures - AUROC and AUPRC. The details of these datasets and performance measures are given in section 7.3.2.

Results

In the experiments, we consider both sequence and GO term based approaches for predicting PPIs. We pick only top few sequence based methods for comparison. These approaches include M1, M2 and M8 (from Chapter 7), that rely on physico-chemical properties of amino acids for feature construction; our representation learning based approach M11 that is entirely data-driven and PIPR, a recent end-to-end deep learning approach. The details of these approaches and results for other sequence based approaches are discussed in detail in Chapter 7.

Table 8.2 and Table 8.3 summarize the performance of each considered method on *human* and *yeast* datasets. For results on *human* datasets, GO term based approaches outperform sequence based methods, specifically for C2 and C3 test cases. When compared among themselves, GO term based approaches perform similarly for the C1 test class. For C2 and C3 test cases, we observe that ULCA performs best among others. We also note that there exists a

gap in performance of representation learning approaches, NREmbd and ULCAEmbd when compared to BOW based approaches NR and ULCA specifically for C2 and C3 test cases. It is important to note here that ULCAEmbd and NREmbd techniques work on low (100) dimensional embeddings for protein pairs which make them computationally attractive when used for making large scale PPI predictions.

Table 8.2: ***Comparison of the prediction performance between the proposed methods and other state of art methods on the*** **human** ***dataset from Dataset1***

Method	AUROC			AUPRC		
	C1	C2	C3	C1	C2	C3
M1 (Martin et al. [143])	0.81 ± 0.01	0.61 ± 0.01	0.58 ± 0.03	0.82 ± 0.01	0.60 ± 0.01	0.57 ± 0.03
M2 (Vert et al. [144])	0.85 ± 0.01	0.60 ± 0.01	0.58 ± 0.02	0.85 ± 0.01	0.60 ± 0.01	0.56 ± 0.03
M8 (Ding et al. [145])	0.82 ± 0.01	0.60 ± 0.02	0.57 ± 0.02	0.83 ± 0.01	0.60 ± 0.01	0.56 ± 0.02
M11 (SuperVecNW)	0.82 ± 0.01	0.58 ± 0.01	0.54 ± 0.02	0.84 ± 0.02	0.59 ± 0.01	0.53 ± 0.01
PIPR	0.80 ± 0.01	0.57 ± 0.01	0.52 ± 0.01	0.80 ± 0.01	0.57 ± 0.01	0.51 ± 0.01
NR	0.81 ± 0.02	0.67 ± 0.01	0.67 ± 0.02	0.82 ± 0.008	0.67 ± 0.005	0.67 ± 0.02
NREmbd	0.80 ± 0.012	0.63 ± 0.005	0.63 ± 0.023	0.82 ± 0.007	0.64 ± 0.004	0.63 ± 0.008
ULCA	0.80 ± 0.006	0.68 ± 0.01	0.68 ± 0.012	0.82 ± 0.003	0.69 ± 0.004	0.68 ± 0.01
ULCAEmbd	0.80 ± 0.009	0.66 ± 0.012	0.65 ± 0.02	0.81 ± 0.005	0.65 ± 0.009	0.65 ± 0.006

Comparing PPI prediction methods on the *yeast* dataset (refer Table 8.3), shows that GO-based methods significantly outperform sequence based approaches on all C1, C2 and C3 test cases. Among the GO-based approaches, similarly to our observation for *human* dataset, ULCA performs better than the other approaches. The performance of ULCAEmbd comes close to NR.

Table 8.3: ***Comparison of the prediction performance between the proposed methods and other state of art methods on the*** **yeast** ***dataset from Dataset1***

Method	AUROC			AUPRC		
	C1	C2	C3	C1	C2	C3
M1 (Martin et al. [143])	0.82 ± 0.01	0.61 ± 0.02	0.58 ± 0.03	0.83 ± 0.01	0.62 ± 0.02	0.57 ± 0.03
M2 (Vert et al. [144])	0.84 ± 0.01	0.60 ± 0.02	0.59 ± 0.03	0.84 ± 0.01	0.61 ± 0.02	0.58 ± 0.03
M8 (Ding et al. [145])	0.82 ± 0.01	0.62 ± 0.02	0.61 ± 0.02	0.84 ± 0.01	0.64 ± 0.02	0.62 ± 0.02
M11 (SuperVecNW)	0.80 ± 0.01	0.58 ± 0.01	0.54 ± 0.01	0.82 ± 0.01	0.59 ± 0.01	0.53 ± 0.02
PIPR	0.84 ± 0.01	0.60 ± 0.02	0.56 ± 0.03	0.84 ± 0.02	0.60 ± 0.02	0.55 ± 0.02
NR	0.87 ± 0.02	0.79 ± 0.01	0.78 ± 0.02	0.89 ± 0.008	0.81 ± 0.005	0.81 ± 0.02
NREmbd	0.86 ± 0.012	0.74 ± 0.005	0.75 ± 0.023	0.88 ± 0.007	0.77 ± 0.004	0.78 ± 0.008
ULCA	0.89 ± 0.006	0.82 ± 0.01	0.82 ± 0.012	0.90 ± 0.003	0.85 ± 0.004	0.85 ± 0.01
ULCAEmbd	0.87 ± 0.009	0.78 ± 0.012	0.79 ± 0.02	0.89 ± 0.005	0.80 ± 0.009	0.81 ± 0.006

To summarize these results, we note that: (i) GO term based approaches gives better results than sequence based methods and hence mays generalise well on the population level. (ii) ULCA features and therefore, ULCA-BOW representations of protein pairs capture the interaction between proteins better

than other representations. (iii) Using embeddings for GO terms in the context of PPI prediction can provide computational advantages for large scale experiments, although there remains scope for improvement in performance.

8.4 Conclusions

In this chapter, we explored the utility of using GO terms for constructing feature vectors for protein pairs and their usage for predicting interactions. For our analysis, we considered BOW based feature construction methods - NR and ULCA. We also presented an approach for learning the representations for GO terms which are further combined to generate a low dimensional representation of protein pairs. With the experimental results we showed that GO terms based methods outperform sequence based methods, specifically for C2 and C3 test cases. We noted that the GO embeddings may prove to be useful for predicting PPI at large scale given their lower dimensionality, while sacrificing some performance.

One of the drawbacks of GO term based methods is the unavailability of these annotations for some proteins, which inhibit such methods for making a prediction for protein pairs that have missing GO annotations. Considering that we can learn useful representations of sequences and GO terms for making PPI prediction opens the possibility of learning joint representations for protein pairs, which can address the limitations associated with these two data sources. Developing such hybrid methods remains future work.

So far, in this thesis, we presented techniques of learning representations for biological sequences. We also showed that these ideas can be extended to learn representations for other entities, e.g. GO terms. Through the experiments discussed in this thesis, we argued that using learned representations for downstream bioinformatics tasks provides some advantages, specifically for applications involving large scale comparisons and predictions.

Representation learning in the bioinformatics domain is a new area, and further opportunities and challenges remain. The work presented in this thesis is a step forward in that direction. In the next chapter, we conclude this thesis, exploring the contributions and limitations of the work we have presented.

We will also consider a number of possible future directions arising from the research undertaken in this dissertation.

9 Conclusions and Further Work

9.1 The Thesis in Review

The primary goal of this thesis is to explore the utility of representation learning approaches for constructing generic or task-specific low dimensional embeddings for biological sequences. We argue that these approaches provide an alternative to current alignment-free methods. We propose new approaches to learn these representations for biological sequences and we demonstrate their utility for various downstream bioinformatics tasks.

Chapter 1 sets the scene of this thesis, highlighting the motivating factors and the objectives to be pursued. Considering that our research lies in the intersection of bioinformatics (focusing on sequence analysis) and machine learning (more specifically representation learning) research, in chapter 2 we reviewed topics relevant to our work from both of these areas. Apart from the general review of these topics, chapter 2 also saw a detailed description of two alignment-based methods - BLAST [25] and MMseqs2 [30] - that we used for benchmarking our work on the *homologous sequence retrieval* task (chapter 5). We then also introduced `Word2Vec` [62] and subsequently the `Doc2Vec` [64] architecture which are fundamental building blocks of our representation learning models.

Having introduced the motivation and context of the research undertaken in this thesis in chapter 1 and chapter 2, in chapter 3, we provided the details of our first representation learning model `Seq2Vec` [153] noting its utility for generating low dimensional yet effective representations for protein sequences. The experiments covered in this chapter demonstrate the superiority of `Seq2Vec` over earlier representation learning approach `BioVec` [20] (section 3.3) on the protein family prediction task. The insights gained from

this work motivated us to work on representation learning frameworks that can further enrich sequence representations.

Subsequently, in chapter 4, we presented new representation learning approaches - `SuperVec` and `SuperVecX` - that provide ways to enrich sequence embeddings by capturing meta-information along with the sequence information in the learned representations. Although training of these models used supervised learning, the resulting embeddings for sequences are generated in a completely unsupervised fashion, i.e., for generating sequence embeddings, the associated meta-information is not required directly. We validated the superiority of these embeddings over those generated using only sequence information for various protein classification (chapter 4) and *homologous sequence retrieval* tasks (chapter 5).

As noted in chapter 5, we focused on the application of the learned sequence representations for *homologous sequence retrieval* tasks. For this purpose, we used nearest neighbourhood-based search in the embedding space. Given the low dimensionality of the embeddings, we were able to achieve a 50 fold reduction in querying time when compared with the standard alignment-based approach - BLAST. We noted that while the direct use of sequence embeddings can provide considerable improvement in querying time, there remains significant scope for improvement in retrieval performance (precision values at different recall levels) when compared to BLAST. We addressed this concern to some extent through the algorithm `H-SuperVec(X)`, that computes a better estimate of sequence similarity - and hence further reduce the performance gap with BLAST. Further, to keep the gain in querying time and improve the precision-recall results, we introduced a hybrid approach (H+BLAST), where `H-SuperVec(X)` is used as a pre-filter to reduce the search space for BLAST. We showed that such a system provides retrieval performance (precision-recall results) comparable to BLAST while still providing substantial reduced querying time.

Having shown the utility of representation learning approaches for many single input problems (e.g. protein family prediction, sequence retrieval), we next focused on exploring their utility for a more complex input, i.e. the Protein-Protein Interaction (PPI) prediction task.

In chapter 6, we provided a detailed review of the PPI problem and the existing experimental and computational approaches. Though different data sources such as gene neighbourhood, sequence data and co-expression data have been investigated by researchers as poteintial inputs, we limit our exploration to approaches based on sequences or GO terms (functional annotations).

First we focused on sequence-based PPI prediction methods in chapter 7 . Most of the available sequence-based methods for predicting PPIs rely on high dimensional feature vectors constructed using physico-chemical properties. Considering the possible computational advantages and their demonstrated utility for single entity problems, we proposed to employ sequence embeddings for predicting PPIs. We conducted extensive experiments for the *human* and *yeast* datasets, following the evaluation strategy suggested by Park et al. [138], that advocate to create three test sets: C1, C2 and C3 - based on the presence or absence of test sample proteins in the training set (refer chapter 6). The results obtained on these datasets suggest that sequence embeddings based approaches could be a better choice than other benchmarked methods, given their computational advantages and competitive performance. One of the drawbacks we noted with sequence-based approaches is that they perform well for the test set C1 but relatively poorly on the more challenging test sets - C2 and C3 - reflecting their inability to generalise at the population level.

Considering the limitation of current sequence-based approaches, in chapter 8 we explored the potential use of an alternative data source ,i.e. GO terms for predicting PPIs. The current methods that utilise GO terms usually rely on a bag-of-words (BOW) based approach for constructing a feature vector for a protein pair. These approaches mostly vary based on the methodology adopted to get GO terms for a protein pair from the GO-DAG. Considering the potential computational advantages, we proposed to learn low dimensional embeddings for GO terms that can be further combined to generate representations for protein pairs. Our results from this study are encouraging, although there remains a scope of improvement regarding the performance compared to BOW-based representations. We demonstrated that

the GO terms based approaches generalise better than sequence-based approaches, but the unavailability of GO terms for many proteins restricts the application of such methods across all protein pairs.

A hybrid approach that uses both sequences and GO terms together can help counter the limitation of current sequence or GO term based approach; it remains a work to be considered in future.

9.2 Central Achievements

In this section we examine the central achievements of the thesis with particular reference to the objectives introduced in chapter 1. We have earlier highlighted some of the main contributions of the work in each chapter but here we focus in particular on the following outcomes:

- We have established the usability of representation learning approaches for biological problems by providing new approaches for learning sequence representations. We have demonstrated their use for downstream bioinformatics tasks and shown that ours is successful compared to existing benchmarks.
- We have extended earlier unsupervised architectures and established that inclusion of meta-data associated with sequences through supervised learning of representations can enrich sequence embeddings and lead to better results for the particular tasks at hand. Our methods `SuperVec` and `SuperVecX` provide ways to capture both sequence and meta-information together in learned sequence representations.
- We have further examined improvements in the representations of sequences within a vector space by computing multiple representations via an approach which we called `H-SuperVec(X)`. This work builds on the approaches `SuperVec` and `SuperVecX` to allow us compute our multiple herierchy for representations. We have demonstrated that these approaches produced a more accurate estimate of sequence similarity allowing us to achieve improve performance retrieving homologous sequences from a database of proteins.

- We have achieved an order of magnitude reduction in querying time with embedding-based methods compared to alignment-based approach BLAST. Still, there remains a scope of improvement in regards to retrieval accuracy. We recognise that a hybrid approach that uses an embedding-based method to pre-filter the database before applying a more accurate approach like BLAST can keep advantages from both worlds, i.e. speed and accuracy. We showed that our proposed hybrid approach `H-SuperVec(X)+BLAST` achieves retrieval performance similar to the BLAST while providing a magnitude of reduction in querying time.
- We have demonstrated that representation learning based approaches provide competitive and computationally attractive alternatives for predicting PPIs compared to current benchmark methods.

In the next section, we will look more carefully at the limitations of our work and the possible future direction for research.

9.3 Discussion and Future Work

Over the course of this thesis we have presented a number of representation learning approaches and have shown their applicability for a variety of downstream tasks. However these approaches have their limitations, while some of them have been addressed in the later chapters of the thesis - building on earlier work from chapter 3 and chapter 4 - a number of limitations remain. These potentially provide good avenues for further work.

Broadly, the representation learning approaches we have considered can be categorised as unsupervised or supervised approaches, where the category is based on training mechanism. Our model `Seq2Vec` for learning sequence representations - uses only sequences for training purposes and can be categorised as an unsupervised model. The representations generated through this model can be used as input to ML algorithms for a given task. While `Seq2Vec` provides low dimensional but useful representations for biological sequences, these representations may not be useful for all problems. As described earlier, we introduced supervised methods `SuperVec` and `SuperVecX`

to accommodate additional label information (such as family annotations) to improve the quality of embeddings for a particular task. It is important to mention here that supervision is required only during the training of these approaches; the generation of sequence embeddings through these models remains an unsupervised process.

As noted earlier, `SuperVec(X)` generated embeddings offer a number of advantages over their unsupervised counterparts for *homologous sequence retrieval*. However, their performance deteriorates with an increasing number of classes (refer to chapter 5). We addressed this limitation to some extent by training these models on multiple splits of the data leading to the method `H-SuperVec(X)`. However, this avenue has not been fully explored. Investigating other approaches which may accommodate large numbers of separate classes without substantial deterioration in the performance may be considered a further research direction. These methods also require classes with a sufficient number of samples (sequences) for training purposes; developing approaches that can accommodate smaller classes is an important direction if we are to capture the full landscape of sequence types. Another possibility is to improve the training process of `SuperVec` and related methods. A detailed discussion of these ideas is provided later in the chapter.

One of the key advantages of low dimensional embeddings is the potential to rapidly exclude irrelevant entries from the database for a query search making it a suitable to be used as a pre-processing filter for more accurate algorithms such as BLAST. We have already taken one step in this direction with the use of our hybrid approach `H-SuperVec(X)`+BLAST, but research in this direction is still in its infancy. A few of the potential extensions are discussed later in the chapter.

In the final chapters of the thesis we have explored the utility of representation learning approaches for PPI prediction task. We noted that while sequence-based approaches do not generalise well for population level, GO-based approaches can not be used to predict interactions for all protein pairs given the limited availability of GO annotations. An integrated approach may potentially provide a better alternative to the existing approaches. It is considered in detail below.

In this discussion, we have briefly covered the limitations of our presented approaches and the possible research directions that can be considered for further work. Below we list some of these potential opportunities in detail.

- **Improving training process of SuperVec** As noted in section 4.2.1, in the training process, for each sample (*k-mer*, context, corresponding sequence tag) negative samples (sequences from other classes) are required. Currently, these are randomly selected from each class. A principled way of choosing such negative samples may improve the training process and hence enhance the quality of sequence embeddings. Selecting negative samples based on their relative distance from the positive class (in the vector space) or based on sequence similarity scores (obtained from alignment algorithms such as BLAST) are potential directions that can be further investigated. Note that selecting negative samples based on the sequence similarity score would incur additional computational and memory overhead to the current approaches, but it is a one-time effort.

- **Hybrid-Methods for Sequence Retrieval:** Large scale sequence comparisons remains a core task in the bioinformatics domain. With the increase in sequence database size, there is a need to work on faster and better alternatives. In our exploration of sequence retrieval work, we note in particular that hybrid approaches like `H+BLAST` can provide an alternative to the computationally expensive alignment-based sequence retrieval tools. Further, there exists an opportunity to improve and standardise such hybrid methods. Note that in the hybrid approach, neighbourhood size in the pre-screening step determines the querying time. A focus on providing a robust mechanism to assess the neighbourhood size and how it affects the overall performance needs a detailed analysis and experiments. Also, methods other than BLAST can be explored as a possible post-screening approach.

- **Semi-supervised framework for representation learning:** Acquiring additional labels (such as functional annotations) for biological sequences is generally costly and time consuming. For this reason we do not have

such annotations for many sequences. Supervised representation learning approaches require labels associated with each sequence for training purposes and are not directly applicable to situations in which there is limited availability of annotations. This provides an opportunity to work on semi-supervised representation learning approaches, methods that rely on only a few labels for training the model. We expect that such methods performance would fall in between unsupervised and supervised approaches. However, semi-supervised methods would extend some of the advantages of the label information to a wider group of sequences.

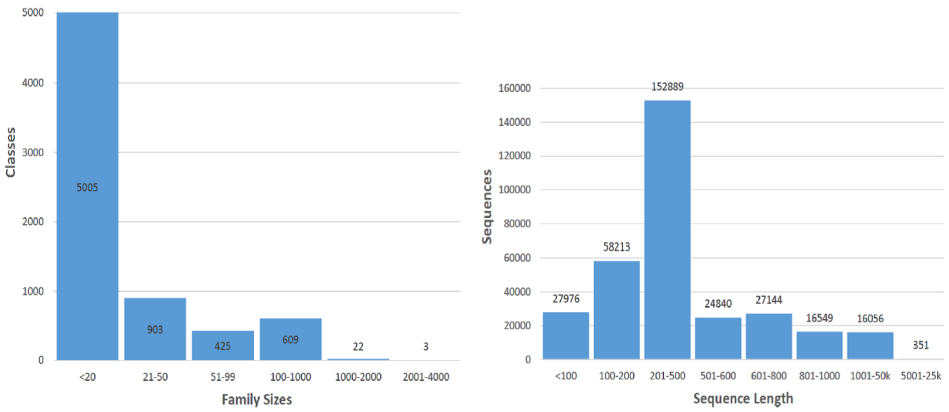
- **An integrated Approach for PPI Prediction:** Based on our analysis in this thesis on PPI prediction problem, we note that available sequence based approaches for PPI prediction do not perform very well for the C2/C3 test cases. Alternatively, GO-term based methods perform significantly better than the sequence based approaches but are limited in their applicability for only the protein pairs for which GO terms are available. Working on an integrated framework that uses sequence and GO terms together for prediction may provide advantages over the existing approaches. Another alternative is to explore the ways to integrate information contained in sequences, network and GO terms for learning better representations for protein pairs. Improving the protein-pair representation is expected to influence the machine learning algorithms resulting in improved results for predicting PPIs.
- **Representation Learning for DNA sequences:** The representation learning approaches presented in this thesis are generic and can be extended to the problems involving DNA sequences. Researchers have recently started exploring unsupervised techniques for this space, and there remains a possibility of improving the quality of DNA sequence representations. Similar to the protein sequences, proposed supervised approaches can be directly adapted to enhance the quality of DNA sequence embeddings by incorporating the relevant meta information. Note that the choice of meta-information depends on the insights and domain knowledge about the problem.

In conclusion, in this work, we have extended the application of representation learning methods for biological sequences. In particular, we have adapted existing approaches, introduced supervised learning approaches and supplemented these by hierarchical methods to work with a large number of classes. Further, we presented hybrid methods that exploit the advantages of embedding based alignment-free methods and alignment-based methods such as BLAST. Such approaches therefore offer improvements in speed while maintaining accuracy, so we have a faster yet accurate alternative for large scale sequence comparisons. Lastly, we showed the potential use of representation learning for more complex input problem such as predicting PPIs.

The applications of representation learning in this domain remain an area of active research, and we have been able to identify some very promising directions for future work.

A Datasets Description

Datset1



(a) Class size distribution (b) Sequence length distribution

Figure A.1: **Distribution of class sizes and lengths of sequences present in the complete dataset.**

Dataset2

In total, there are 33 million representative sequences. For many of these sequences, no PFAM annotation is available. For our purposes, we keep only those sequences that are annotated with single "PFAM" entry, leaving us with 16968 unique "PFAM" entries (classes) containing $87,358,58$ sequences. These classes vary in size, the largest one contains $81,739$, whereas more than $24,00$ classes have less than 10 entries. For experiments, we have to select classes that have a reasonable size to make training and test split we choose the classes that contain $400 - 1000$ sequences. There are 1866 such classes with a total of $11,921,50$ sequences that comprises dataset2.

B Update Equations for SuperVec

The update equations for the parameters - $\mathbf{s}_i$, $\mathbf{k}_{i_j}$, $\mathbf{g}_q$, $\mathbf{s}_r$ and $\mathbf{s}_z$ of `SuperVec` are obtained using gradient descent. As usual, parameter updates rely on their gradient with respect to the objective function. If we want to find the value of $\mathbf{x}$ for which $f(\mathbf{x})$ is minimised, we initialize $\mathbf{x}$ randomly and update its value in each iteration. The update equation for $\mathbf{x}$ is then

$$\mathbf{x}^{new} = \mathbf{x} - \eta \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}, \tag{B.1}$$

where η is the learning rate. For `SuperVec` the overall loss function is given by

$$J(\mathbf{S}, \mathbf{K}) = \sum_{s_i \in \mathcal{S}} \sum_{j=1}^{n_i} \left[\underbrace{-\log \Pr\left[k_{i_j} \mid C_{i_j},\ s_i\right]}_{NN1} - \gamma \sum_{z \in \mathcal{I}_i^+} \underbrace{\log \Pr\left[s_z \mid s_i\right]}_{NN2} \right]. \tag{B.2}$$

For $NN1$ we employ a hierarchical softmax approach and for $NN2$ we use a negative sampling technique. Hierarchical softmax (HS) is based on a binary tree ($\mathcal{T}$) construction, where each leaf maps to a *k-mer* in $\mathcal{K}$; also, since it is a binary tree, any leaf (*k-mer*) can be reached rapidly from the root by traversing a unique path. We denote the q^{th} node in such a path from the root to the leaf *k-mer* k_{i_j} as $n(k_{i_j}, q)$ and the corresponding vector as $\mathbf{g}_{k_{i_j},q}$. Considering a stack of these internal node vectors of $\mathcal{T}$ yields the internal node matrix, $\mathbf{G}$. The probability for a specific *k-mer* k_{i_j} is obtained by traversing from the root to the leaf corresponding to k_{i_j} and multiplying the contributions (defined below) from each internal node in the path. When calculated using the hierarchical

softmax approach, $\Pr\left[k_{i_j} \mid C_{i_j},\ s_i\right]$ may then be written

$$\prod_{q=1}^{L_{i_j}-1} \sigma\left(child(q-1)\cdot\langle \mathbf{g}_{k_{i_j},q}, \mathbf{h}_{i_j}\rangle\right), \tag{B.3}$$

where $child(q-1) = 1$ if $n(k_{i_j}, q)$ is the left child of $n(k_{i_j}, q-1)$ and -1 otherwise, and L_{i_j} is the length of the path i.e. the total number of nodes in the path from the root to the j^{th} *k-mer* of sequence s_i; $\mathbf{h}_{i_j}$ is the sum of vectors corresponding to context C_{i_j} and sequence s_i. Finally, the overall objective function is written as,

$$\begin{aligned}\mathrm{J}^{'}(\mathbf{S},\mathbf{K},\mathbf{G}) = \min \sum_{s_i\in\mathcal{S}}\sum_{j=1}^{n_i}\sum_{q=1}^{L_{i_j}-1}\Bigg[&-\log\ \sigma\left(child(q-1)\cdot\langle \mathbf{g}_{k_{i_j},q}, \mathbf{h}_{i_j}\rangle\right) \\ &-\gamma\sum_{z\in\mathcal{I}_i^+}\left(\log\sigma\left(\langle \mathbf{s}_z, \mathbf{s}_i\rangle\right) + \sum_{r\in\mathcal{I}_i^-}\log\sigma\left(-\langle \mathbf{s}_r, \mathbf{s}_i\rangle\right)\right)\Bigg].\end{aligned} \tag{B.4}$$

The update equation for `SuperVec` parameters are obtained by differentiating Eq (B.4) w.r.t the parameter and takes the same form as given for $\mathbf{x}$ in Eq (B.1). For example, the update equation for parameter $\mathbf{s}_i$ is given as:

$$\mathbf{s}_i^{new} = \mathbf{s}_i - \eta\frac{\partial \mathrm{J}^{'}(\mathbf{S},\mathbf{K},\mathbf{G})}{\partial \mathbf{s}(i)}. \tag{B.5}$$

For notational convenience, we write $\mathrm{J}^{'}(\mathbf{S},\mathbf{K},\mathbf{G})$, $\mathbf{g}_{k_{i_j},q}$, $\mathbf{h}_{i_j}$, $child(q-1)$ as $\mathrm{J}^{'}$, $\mathbf{g}_q$, $\mathbf{h}$ and $[\![q-1]\!]$ respectively. The derivation of gradient of parameters w.r.t $\mathrm{J}^{'}$ are given below:

$$\frac{\partial \mathrm{J}^{'}}{\partial \mathbf{s}_i} = \sum_{q=1}^{L_{i_j}-1} \Bigg[\frac{-\partial \log\ \sigma([\![q-1]\!] \cdot \langle \mathbf{g}_q, \mathbf{h} \rangle)}{\partial \langle \mathbf{g}_q, \mathbf{h} \rangle} \cdot \frac{\partial \langle \mathbf{g}_q, \mathbf{h} \rangle}{\partial \mathbf{s}_i} - \gamma \sum_{z \in \mathcal{I}_i^+} \left(\frac{\partial \log \sigma\left(\langle \mathbf{s}_z, \mathbf{s}_i \rangle\right)}{\partial \langle \mathbf{s}_z, \mathbf{s}_i \rangle} \cdot \frac{\partial \langle \mathbf{s}_z, \mathbf{s}_i \rangle}{\partial \mathbf{s}_i} + \sum_{r \in \mathcal{I}_i^-} \frac{\partial \log \sigma\left(-\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right)}{\partial \left(-\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right)} \cdot \frac{\partial \left(-\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right)}{\partial \mathbf{s}_i} \right) \Bigg]. \tag{B.6}$$

using the identities:

$$\begin{aligned} \frac{\partial \log \sigma\left(\mathbf{x}\right)}{\partial \left(\mathbf{x}\right)} &= \sigma\left(-\mathbf{x}\right) \\ \frac{\partial \log \sigma\left(\text{-}\mathbf{x}\right)}{\partial \left(\text{-}\mathbf{x}\right)} &= \sigma\left(\mathbf{x}\right) \end{aligned} \tag{B.7}$$

$\frac{\partial \mathrm{J}^{'}}{\partial \mathbf{s}_i}$ is simplified as:

$$\frac{\partial \mathrm{J}^{'}}{\partial \mathbf{s}_i} = -\sum_{q=1}^{L_{i_j}-1} \Bigg[\sigma(-[\![q-1]\!] \cdot \langle \mathbf{g}_q, \mathbf{h} \rangle) [\![q-1]\!] \cdot \mathbf{g}_q + \gamma \sum_{z \in \mathcal{I}_i^+} \left(\sigma\left(-\langle \mathbf{s}_z, \mathbf{s}_i \rangle\right) \cdot \mathbf{s}_z + \sum_{r \in \mathcal{I}_i^-} -\sigma\left(\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right) \cdot \mathbf{s}_r \right) \Bigg]. \tag{B.8}$$

further, using the identity:

$$\sigma\left(-\mathbf{x}\right) = 1 - \sigma\left(\mathbf{x}\right) \tag{B.9}$$

we can write first part of Eq (B.8) as,

$$\Big[\sigma(-[\![q-1]\!]\cdot\langle\mathbf{g}_q,\mathbf{h}\rangle)\Big]\,[\![q-1]\!] = \Big[1-\sigma([\![q-1]\!]\cdot\langle\mathbf{g}_q,\mathbf{h}\rangle)\Big]\,[\![q-1]\!]. \tag{B.10}$$

As noted before, $[\![q-1]\!]$ takes value 1 or -1. Eq (B.10) can be further simplified as:

$$\Big[1-\sigma([\![q-1]\!]\cdot\langle\mathbf{g}_q,\mathbf{h}\rangle)\Big]\,[\![q-1]\!] = \begin{cases} 1-\sigma\left(\langle\mathbf{g}_q,\mathbf{h}\rangle\right) & if\ [\![q-1]\!]=1, \\ -\sigma\left(\langle\mathbf{g}_q,\mathbf{h}\rangle\right) & if\ [\![q-1]\!]=-1 \end{cases}$$

$$= t-\left(\sigma\left(\langle\mathbf{g}_q,\mathbf{h}\rangle\right)\right) \tag{B.11}$$

where $t=1$ if $[\![q-1]\!]=1$ and 0 otherwise.

Using Eq (B.11), we write the final expression for $\frac{\partial \mathrm{J}'}{\partial \mathbf{s}_i}$ as,

$$\frac{\partial \mathrm{J}'}{\partial \mathbf{s}_i} = -\sum_{q=1}^{L_{i_j}-1}\Bigg[t-\left(\sigma\left(\langle\mathbf{g}_q,\mathbf{h}\rangle\right)\right) + \gamma\sum_{z\in\mathcal{I}_i^+}\left(\sigma\left(-\langle\mathbf{s}_z,\mathbf{s}_i\rangle\right)\cdot\mathbf{s}_z+\sum_{r\in\mathcal{I}_i^-}-\sigma\left(\langle\mathbf{s}_r,\mathbf{s}_i\rangle\right)\cdot\mathbf{s}_r\right)\Bigg]. \tag{B.12}$$

Using Eq (B.8) and Eq (B.11) the gradients for the parameters: $\mathbf{k}_{i_j}$, $\mathbf{g}_q$, $\mathbf{s}_z$, $\mathbf{s}_r$ is given as below.

$$\begin{aligned}\frac{\partial \mathrm{J}'}{\partial \mathbf{k}_{i_j}} &= \sum_{q=1}^{L_{i_j}-1}\frac{-\partial\log\ \sigma([\![q-1]\!]\cdot\langle\mathbf{g}_q,\mathbf{h}\rangle)}{\partial\langle\mathbf{g}_q,\mathbf{h}\rangle}\cdot\frac{\partial\langle\mathbf{g}_q,\mathbf{h}\rangle}{\partial\mathbf{k}_{i_j}} \\ &= -\sum_{q=1}^{L_{i_j}-1} t-\sigma(\langle\mathbf{g}_q,\mathbf{h}\rangle)\cdot\mathbf{g}_q\end{aligned} \tag{B.13}$$

$$\begin{aligned}\frac{\partial \mathrm{J}^{'}}{\partial \mathbf{g}_q} &= \sum_{q=1}^{L_{i_j}-1} \frac{-\partial \log\ \sigma([\![q-1]\!] \cdot \langle \mathbf{g}_q, \mathbf{h} \rangle)}{\partial \langle \mathbf{g}_q, \mathbf{h} \rangle} \cdot \frac{\partial \langle \mathbf{g}_q, \mathbf{h} \rangle}{\partial \mathbf{g}_q} \\ &= -\sum_{q=1}^{L_{i_j}-1} t - \sigma(\langle \mathbf{g}_q, \mathbf{h} \rangle) \cdot \mathbf{h}\end{aligned} \tag{B.14}$$

$$\begin{aligned}\frac{\partial \mathrm{J}^{'}}{\partial \mathbf{s}_z} &= -\gamma \frac{\partial \log \sigma\left(\langle \mathbf{s}_z, \mathbf{s}_i \rangle\right)}{\partial \langle \mathbf{s}_z, \mathbf{s}_i \rangle} \cdot \frac{\partial \langle \mathbf{s}_z, \mathbf{s}_i \rangle}{\partial \mathbf{s}_z} \\ &= -\gamma \sigma\left(-\langle \mathbf{s}_z, \mathbf{s}_i \rangle\right) \cdot \mathbf{s}_i\end{aligned} \tag{B.15}$$

$$\begin{aligned}\frac{\partial \mathrm{J}^{'}}{\partial \mathbf{s}_r} &= -\gamma \sum_{z \in \mathcal{I}_i^+} \frac{\partial \log \sigma\left(-\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right)}{\partial \left(-\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right)} \cdot \frac{\partial \left(-\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right)}{\partial \mathbf{s}_r} \\ &= \gamma \sum_{z \in \mathcal{I}_i^+} \sigma\left(\langle \mathbf{s}_r, \mathbf{s}_i \rangle\right) \cdot \mathbf{s}_i\end{aligned} \tag{B.16}$$

C Herierchical Split Selection

In this experiment, we conduct a retrieval task on randomly selected database (of eight classes) from dataset1 to analyze the effect of partition on `H-SuperVec` performance. We compare the precision-recall values obtained by employing `H-SuperVec` against `SuperVec` and report the percentage change in the precision value at different recall levels. The results obtained from all 35 equal sized partitions of eight classes show a similar trend. The results for five of these partitioned is provided in Fig C.1. These results demonstrate that:

- The choice of the partition at the root node has a negligible impact on the performance of `H-SuperVec`; in other words, a random partition may be chosen for applying `H-SuperVec`.

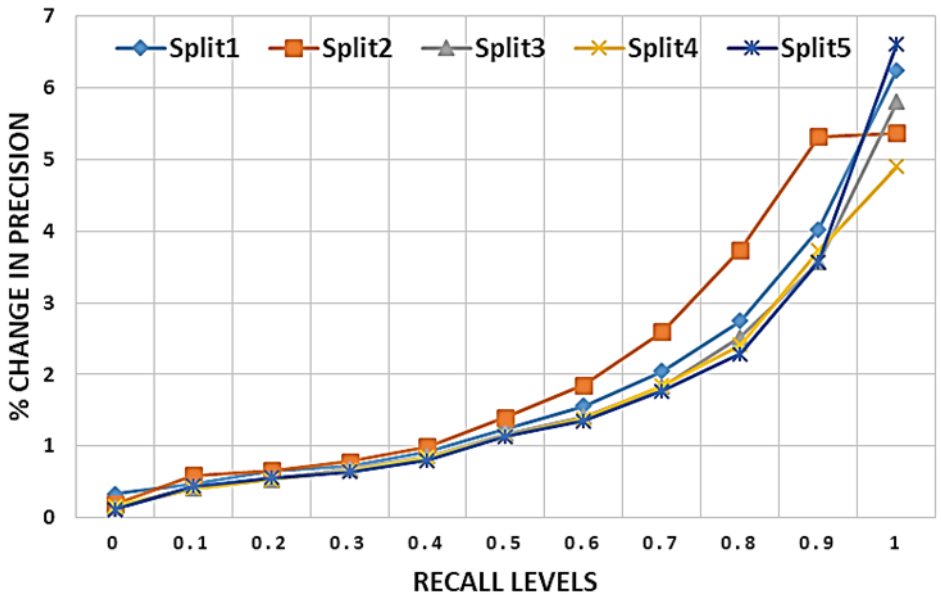


Figure C.1: **Percentage change in precision values:** ***These plots shows the percentage change in precision values obtained for herierchical approach (`H-SuperVec`) as compared to `SuperVec` for the retrieval task performed on randomly chosen eight classes. The graph is shown for five splits among possible 35 splits.***

D Comparison of all methods on retrieval task

In Fig D.1, we provide the comparison of all methods on retrieval task on a database of $\sim 90k$ sequences; the results are averaged over $\sim 60k$ queries. As shown, the proposed supervised approaches provide a high gain in precision values over unsupervised methods and comparable performance as compared to BLAST.

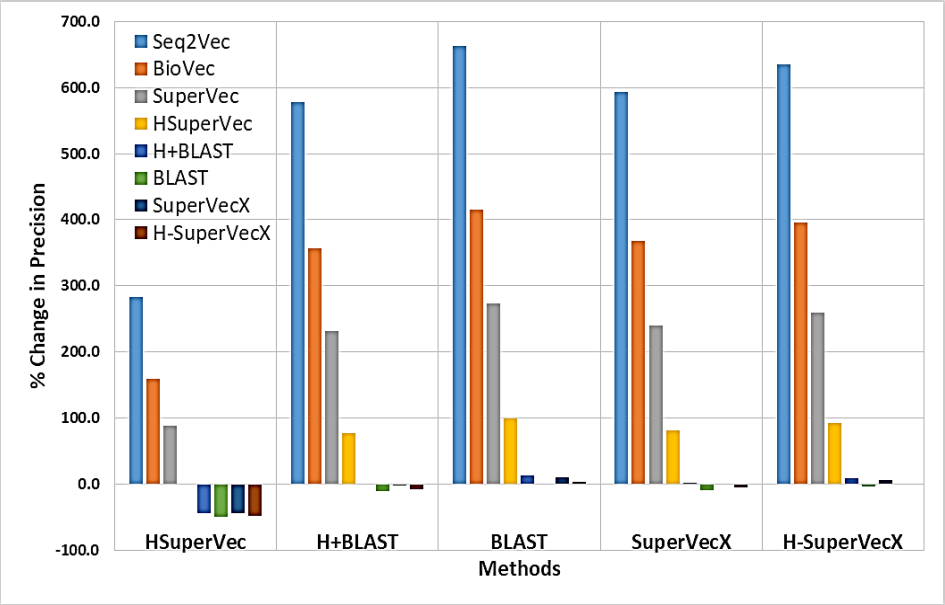


Figure D.1: **Improvement in precision***: The plot shows the percentage improvement in precision value achieved by* `SuperVec`*,* `HSuperVec(X)` *and* `H+BLAST` *over other methods. Precision is compared at* 0.6 *recall for the database of* 90k *sequences and 200 classes from dataset1.*

E Enlarged T-SNE plots for various Embedding Techniques

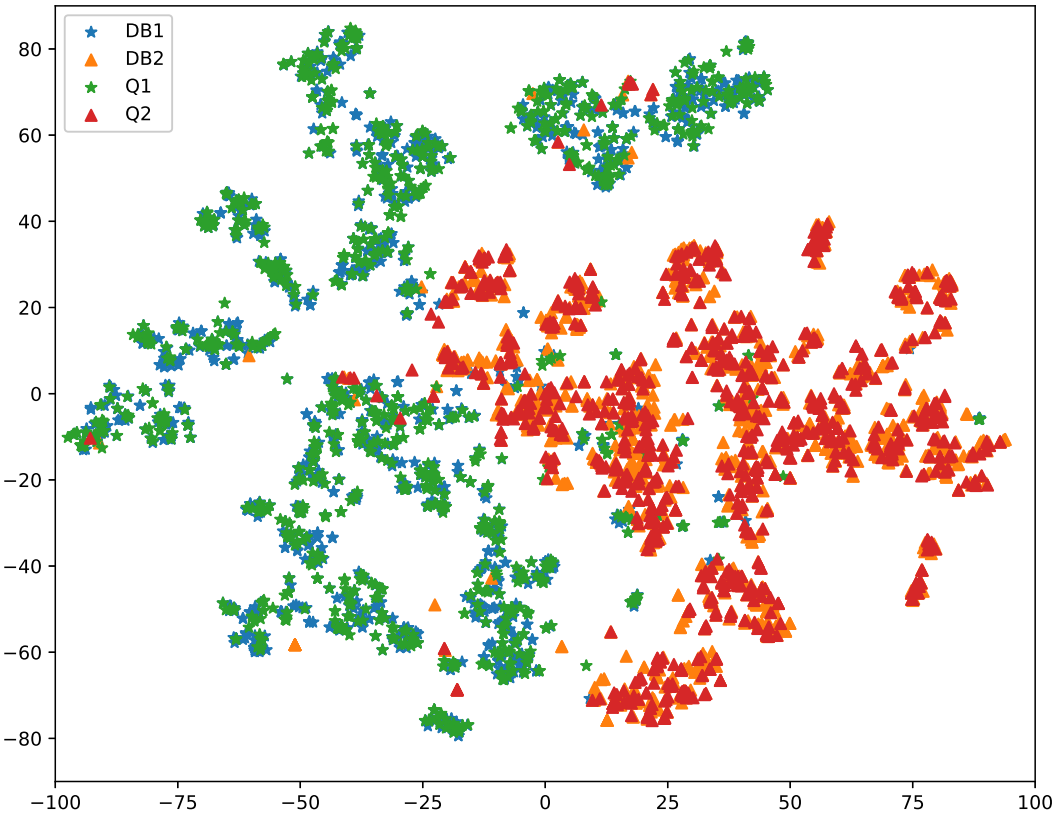


Figure E.1: ***Enlarged T-SNE plot: BioVec***

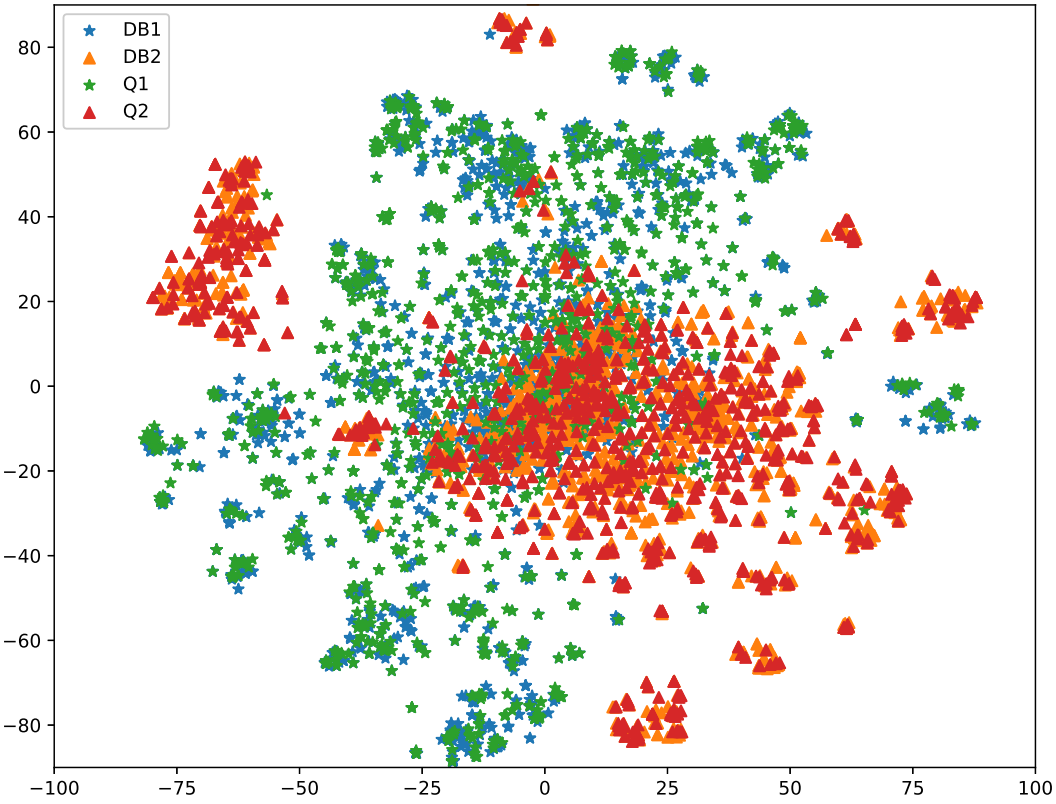


Figure E.2: ***Enlarged T-SNE plot: Seq2Vec***

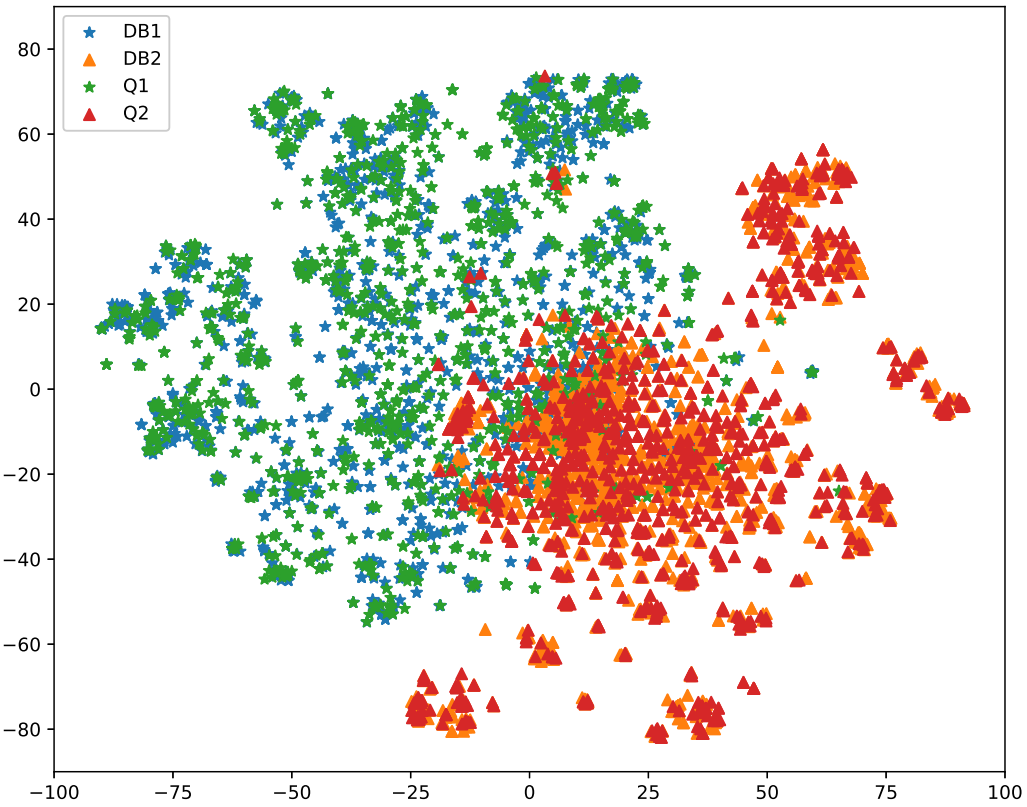


Figure E.3: ***Enlarged T-SNE plot: SuperVec***

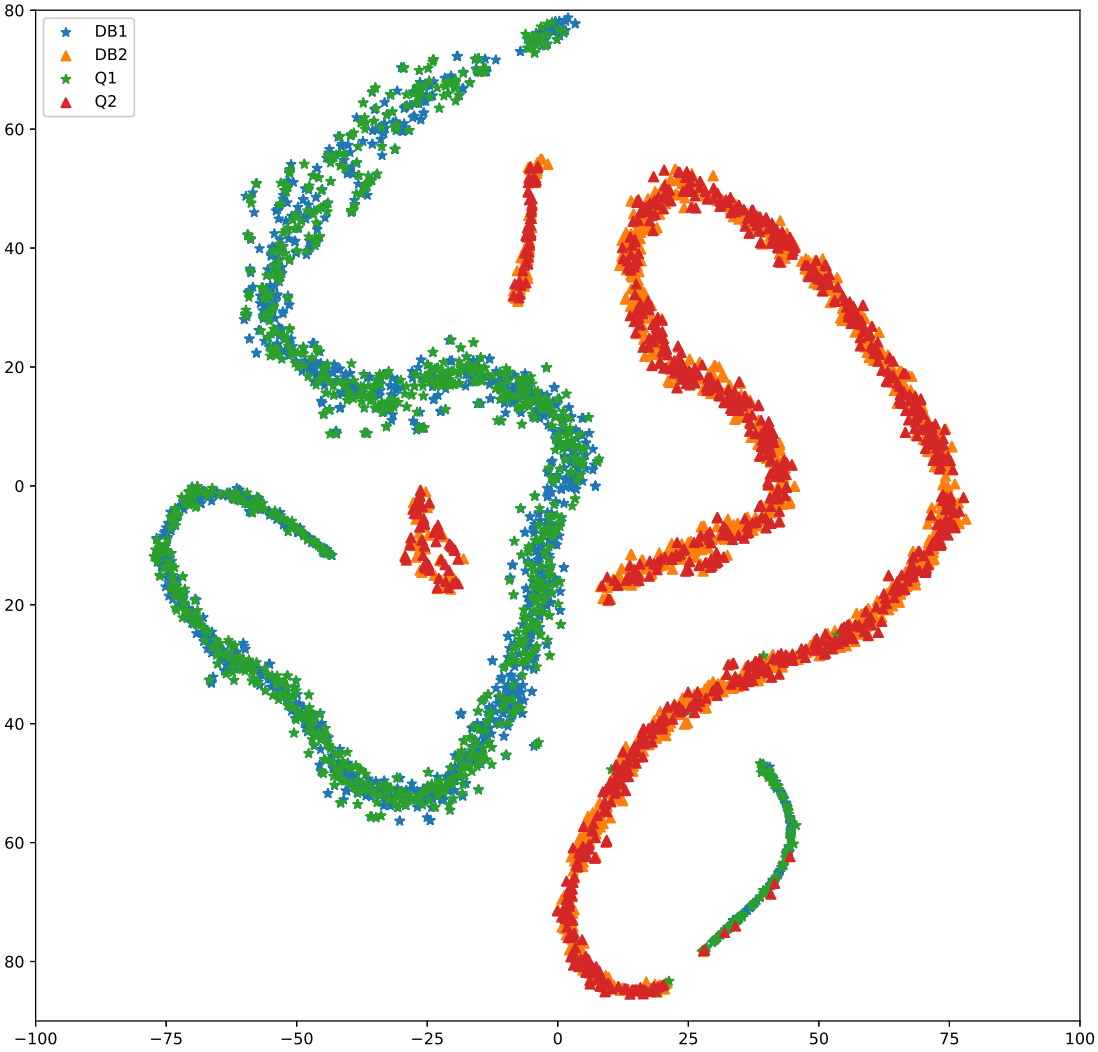


Figure E.4: ***Enlarged T-SNE plot: SuperVecX***

Bibliography

[1] Andrzej Zielezinski et al. "Benchmarking of alignment-free sequence comparison methods". In: *Genome biology* 20.1 (2019), pp. 1–18.

[2] Susana Vinga and Jonas Almeida. "Alignment-free sequence comparison—a review". In: *Bioinformatics* (2003), pp. 513–523.

[3] Andrzej Zielezinski et al. "Alignment-free sequence comparison: benefits, applications, and tools". In: *Genome biology* (2017), p. 186.

[4] Nobuyoshi Nagamine and Yasubumi Sakakibara. "Statistical prediction of protein–chemical interactions based on chemical structure and mass spectrometry data". In: *Bioinformatics* (2007), pp. 2004–2012.

[5] Manfred Koegl and Peter Uetz. "Improving yeast two-hybrid screening systems". In: *Briefings in Functional Genomics and Proteomics* (2007), pp. 302–312.

[6] S B Needleman and C D Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." In: *Journal of molecular biology* (Mar. 1970), pp. 443–53.

[7] T F Smith and M S Waterman. "Identification of common molecular subsequences." In: *Journal of molecular biology* (Mar. 1981), pp. 195–197.

[8] William R Pearson. "Using the FASTA program to search protein and DNA sequence databases". In: *Computer Analysis of Sequence Data*. Springer, 1994, pp. 307–331.

[9] Julie D Thompson et al. "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice". In: *Nucleic acids research* 22.22 (1994), pp. 4673–4680.

[10] Robert C Edgar. "MUSCLE: multiple sequence alignment with high accuracy and high throughput". In: *Nucleic acids research* 32.5 (2004), pp. 1792–1797.

[11] Kazutaka Katoh et al. "MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform". In: *Nucleic acids research* 30.14 (2002), pp. 3059–3066.

[12] Stephen F Altschul et al. "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs". In: *Nucleic acids research* 25.17 (1997), pp. 3389–3402.

[13] SR Eddy et al. "HMMER-biosequence analysis using profile hidden Markov models". In: *URL http://hmmer. janelia. org* (2007).

[14] Aaron CE Darling et al. "Mauve: multiple alignment of conserved genomic sequence with rearrangements". In: *Genome research* 14.7 (2004), pp. 1394–1403.

[15] Scott Schwartz et al. "Human–mouse alignments with BLASTZ". In: *Genome research* 13.1 (2003), pp. 103–107.

[16] Mathieu Blanchette et al. "Aligning multiple genomic sequences with the threaded blockset aligner". In: *Genome research* 14.4 (2004), pp. 708–715.

[17] Michael R Garey and David S Johnson. *Computers and intractability*. Vol. 174. freeman San Francisco, 1979.

[18] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of the International Conference on Learning Representations (ICLR 2013)* (2013), pp. 1–12.

[19] Qv Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *International Conference on Machine Learning - ICML 2014* (2014), pp. 1188–1196.

[20] Ehsaneddin Asgari and Mohammad R. K. Mofrad. "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics". In: *PLOS ONE* (Nov. 2015).

[21] Gene Ontology Consortium et al. "Expansion of the Gene Ontology knowledgebase and resources". In: *Nucleic acids research* (2017), pp. D331–D338.

[22] David J Lipman and William R Pearson. "Rapid and sensitive protein similarity searches". In: *Science* (1985), pp. 1435–1441.

[23] Margaret O Dayhoff. "Atlas of protein sequence and structure". In: (1965).

[24] William R Pearson. "Effective protein sequence comparison". In: *Methods in enzymology* 266 (1996), pp. 227–258.

[25] Steven Henikoff and Jorja G Henikoff. "Amino acid substitution matrices from protein blocks". In: *Proceedings of the National Academy of Sciences* (1992), pp. 10915–10919.

[26] William R Pearson. "Comparison of methods for searching protein sequence databases". In: *Protein Science* (1995), pp. 1145–1160.

[27] Eugene G Shpaer et al. "Sensitivity and selectivity in protein similarity searches: a comparison of Smith–Waterman in hardware to BLAST and FASTA". In: *Genomics* (1996), pp. 179–191.

[28] W James Kent. "BLAT—the BLAST-like alignment tool". In: *Genome research* (2002), pp. 656–664.

[29] Robert C Edgar. "Search and clustering orders of magnitude faster than BLAST". In: *Bioinformatics* (2010), pp. 2460–2461.

[30] Martin Steinegger and Johannes Söding. "MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets". In: *Nature biotechnology* (2017), p. 1026.

[31] Michael L Metzker. "Sequencing technologies [mdash] the next generation". In: *Nat Rev Genet* (2010), pp. 31–46.

[32] Kai Song et al. "New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing." In: *Briefings in bioinformatics* (May 2014), pp. 343–53.

[33] B Edwin Blaisdell. "A measure of the similarity of sets of sequences not requiring sequence alignment". In: *Proceedings of the National Academy of Sciences* (1986), pp. 5155–5159.

[34] Susana Vinga. "Information theory applications for biological sequence analysis". In: *Briefings in bioinformatics* (2013), pp. 376–389.

[35] H Joel Jeffrey. "Chaos game representation of gene structure". In: *Nucleic acids research* (1990), pp. 2163–2170.

[36] Jonas S Almeida. "Sequence analysis by iterated maps, a review". In: *Briefings in bioinformatics* (2013), pp. 369–375.

[37] Jia Wen and YuYan Zhang. "A 2D graphical representation of protein sequence and its numerical characterization". In: *Chemical Physics Letters* (2009), pp. 281–286.

[38] Milan Randić et al. "Unique graphical representation of protein sequences based on nucleotide triplet codons". In: *Chemical Physics Letters* (2004), pp. 247–252.

[39] Igor Ulitsky et al. "The average common substring approach to phylogenomic reconstruction". In: *Journal of Computational Biology* (2006), pp. 336–350.

[40] Bernhard Haubold et al. "Genome comparison without alignment using shortest unique substrings". In: *BMC bioinformatics* (2005), p. 123.

[41] Armando J Pinho et al. "On finding minimal absent words". In: *BMC bioinformatics* (2009), p. 137.

[42] Lianping Yang et al. "Large local analysis of the unaligned genome and its application". In: *Journal of Computational Biology* (2013), pp. 19–29.

[43] Michael Höhl et al. "Pattern-based phylogenetic distance estimation and tree reconstruction". In: *Evolutionary Bioinformatics* (2006), pp. 11–76.

[44] Michael Höhl and Mark A Ragan. "Is multiple-sequence alignment required for accurate inference of phylogeny?" In: *Systematic Biology* (2007), pp. 206–221.

[45] G. E. Sims et al. "Whole-genome phylogeny of mammals: Evolutionary information in genic and nongenic regions". In: *Proceedings of the National Academy of Sciences* (Oct. 2009), pp. 17077–17082.

[46] Ying Wang et al. "Comparison of metatranscriptomic samples based on k-tuple frequencies". In: *PLoS One* (2014), p. 84348.

[47] Tiee-Jian Wu et al. "Optimal word sizes for dissimilarity measures and estimation of the degree of dissimilarity between DNA sequences". In: *Bioinformatics* (2005), pp. 4125–4132.

[48] Oliver Bonham-Carter et al. "Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis." In: *Briefings in bioinformatics* (Nov. 2014), pp. 890–905.

[49] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications. Texts in Computer Science*. Vol. 9. Springer, New York, 2008.

[50] Hasan H Otu and Khalid Sayood. "A new sequence distance measure for phylogenetic tree construction". In: *Bioinformatics* (2003), pp. 2122–2130.

[51] Ming Li et al. "The similarity metric". In: *IEEE transactions on Information Theory* (2004), pp. 3250–3264.

[52] George Dahl et al. "Phone recognition with the mean-covariance restricted Boltzmann machine". In: *Advances in neural information processing systems*. 2010, pp. 469–477.

[53] Frank Seide et al. "Feature engineering in context-dependent deep neural networks for conversational speech transcription". In: *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE. 2011, pp. 24–29.

[54] Geoffrey Hinton et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal Processing Magazine* (2012), pp. 82–97.

[55] Dan Ciregan et al. "Multi-column deep neural networks for image classification". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 3642–3649.

[56] Salah Rifai et al. "The manifold tangent classifier". In: *Advances in Neural Information Processing Systems*. 2011, pp. 2294–2302.

[57] Alex Krizhevsky et al. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[58] Yoshua Bengio. "Neural net language models". In: *Scholarpedia* (2008), p. 3881.

[59] Richard Socher et al. "Semi-supervised recursive autoencoders for predicting sentiment distributions". In: *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics. 2011, pp. 151–161.

[60] Nitish Srivastava and Ruslan R Salakhutdinov. "Multimodal learning with deep boltzmann machines". In: *Advances in neural information processing systems*. 2012, pp. 2222–2230.

[61] Yoshua Bengio et al. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* (2013), pp. 1798–1828.

[62] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[63] J. R. Firth. "A synopsis of linguistic theory 1930-55." In: 1952-59 (1957), pp. 1–32.

[64] Quoc V Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents." In: *ICML*. 2014, pp. 1188–1196.

[65] Ryan Kiros et al. "Skip-thought vectors". In: *Advances in neural information processing systems*. 2015, pp. 3294–3302.

[66] Kai Sheng Tai et al. "Improved semantic representations from tree-structured long short-term memory networks". In: *arXiv preprint arXiv:1503.00075* (2015).

[67] Ian Goodfellow et al. *Deep Learning*. MIT Press, 2016.

[68] Dhananjay Kimothi et al. "Distributed Representations for Biological Sequence Analysis". In: *arXiv preprint arXiv:1608.05949* (2016).

[69] Jingcheng Du et al. "Gene2Vec: Distributed Representation of Genes Based on Co-Expression". In: *bioRxiv* (2018), p. 286096.

[70] Aparajita Dutta et al. "SpliceVec: distributed feature representations for splice junction prediction". In: *Computational biology and chemistry* (2018), pp. 434–441.

[71] Damianos P. Melidis et al. *dom2vec: Unsupervised protein domain embeddings capture domains structure and function providing data-driven insights into collocations in domain architectures*. en. preprint. Bioinformatics, Mar. 2020. (Visited on 06/15/2020).

[72] Patrick Ng. "dna2vec: Consistent vector representations of variable-length k-mers". In: *arXiv preprint arXiv:1701.06279* (2017).

[73] Tristan Bepler and Bonnie Berger. "Learning protein sequence embeddings using information from structure". In: *arXiv preprint arXiv:1902.08661* (2019).

[74] Nils Strodthoff et al. "UDSMProt: universal deep sequence models for protein classification". en. In: *Bioinformatics* 36 (Apr. 2020). Ed. by Yann Ponty, pp. 2401–2409.

[75] Michael Heinzinger et al. *Modeling the language of life – Deep Learning Protein Sequences*. en. preprint. Bioinformatics, Apr. 2019. (Visited on 06/15/2020).

[76] Hesham ElAbd et al. "Amino acid encoding for deep learning applications". en. In: *BMC Bioinformatics* 1 (Dec. 2020), p. 235. (Visited on 06/15/2020).

[77] Frederic Morin and Yoshua Bengio. "Hierarchical Probabilistic Neural Network Language Model." In: *Aistats*. Vol. 5. Citeseer. 2005, pp. 246–252.

[78] MO Dayhoff. "Computer analysis of protein sequences". In: *Computers in Life Science Research*. Springer, 1974, pp. 9–14.

[79] MO Dayhoff et al. "Evolution of sequences within protein superfamilies". In: *Naturwissenschaften* (1975), pp. 154–161.

[80] Loredana Lo Conte et al. "SCOP: a structural classification of proteins database". In: *Nucleic acids research* (2000), pp. 257–259.

[81] A J Enright et al. "An efficient algorithm for large-scale detection of protein families." In: *Nucleic acids research* (Apr. 2002), pp. 1575–84.

[82] P Bork et al. "Predicting function: from genes to genomes and back." In: *Journal of molecular biology* 283 (Nov. 1998), pp. 707–25. ISSN: 0022-2836.

[83] Michael Remmert et al. "HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment". In: *Nature Methods* (Dec. 2011), pp. 173–175.

[84] UniProt Consortium et al. "UniProt: a hub for protein information". In: *Nucleic acids research* (2014).

[85] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[86] Gene Ontology Consortium. "The Gene Ontology (GO) database and informatics resource". In: *Nucleic acids research* (2004), pp. D258–D261.

[87] Robert D Finn et al. "The Pfam protein families database: towards a more sustainable future". In: *Nucleic acids research* (2015), pp. D279–D285.

[88] Alexey G Murzin et al. "SCOP: a structural classification of proteins database for the investigation of sequences and structures". In: *Journal of molecular biology* (1995), pp. 536–540.

[89] Daniel Christopher Esposito et al. "Inferring edge function in protein-protein interaction networks". In: *bioRxiv* (2018), p. 321984.

[90] Armand Joulin et al. "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759* (2016).

[91] Vladimir Zolotov and David Kung. "Analysis and optimization of fastText linear text classifier". In: *arXiv preprint arXiv:1702.05531* (2017).

[92] Ranko Gacesa et al. "Machine learning can differentiate venom toxins from other proteins having non-toxic physiological functions". In: *PeerJ Computer Science* (2016), e90.

[93] Olof Emanuelsson et al. "Locating proteins in the cell using TargetP, SignalP and related tools". In: *Nature protocols* (2007), p. 953.

[94] Yu Li et al. "DEEPre: sequence-based enzyme EC number prediction by deep learning". In: *Bioinformatics* (2017), pp. 760–769.

[95] Ehsaneddin Asgari et al. "Probabilistic variable-length segmentation of protein sequences for discriminative motif discovery (DiMotif) and sequence embedding (ProtVecX)". In: *Scientific reports* (2019), p. 3577.

[96] Baris E Suzek et al. "UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches". In: *Bioinformatics* (2014), pp. 926–932.

[97] Jeff Johnson et al. "Billion-scale similarity search with GPUs". In: *arXiv preprint arXiv:1702.08734* (2017).

[98] Christopher D. Manning et al. In: *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. Chap. 8, pp. 158–159. ISBN: 0521865719, 9780521865715.

[99] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* (2008), pp. 2579–2605.

[100] Javier De Las Rivas and Celia Fontanillo. "Protein–protein interactions essentials: key concepts to building and analyzing interactome networks". In: *PLoS computational biology* (2010), e1000807.

[101] Rui-Sheng Wang et al. "Analysis on multi-domain cooperation for predicting protein-protein interactions". In: *BMC bioinformatics* (2007), p. 391.

[102] Haiyuan Yu et al. "High-quality binary protein interaction map of the yeast interactome network". In: *Science* (2008), pp. 104–110.

[103] Luke Hakes et al. "Protein interactions from complexes: a structural perspective". In: *Comparative and functional genomics* ().

[104] Benjamin A Shoemaker and Anna R Panchenko. "Deciphering protein–protein interactions. Part II. Computational methods to predict protein and domain interaction partners". In: *PLoS computational biology* 3.4 (2007), e43.

[105] Stanley Fields and Ok-kyu Song. "A novel genetic system to detect protein–protein interactions". In: *Nature* (1989), p. 245.

[106] Dong-Soo Han et al. "PreSPI: a domain combination based prediction system for protein–protein interaction". In: *Nucleic acids research* (2004), pp. 6312–6320.

[107] Kyu Kim Wan et al. "Large scale statistical prediction of protein-protein interaction by potentially interacting domain (PID) pair". In: *Genome Informatics* (2002), pp. 42–50.

[108] Einat Sprinzak and Hanah Margalit. "Correlated sequence-signatures as markers of protein-protein interaction". In: *Journal of molecular biology* (2001), pp. 681–692.

[109] Utkan Ogmen et al. "PRISM: protein interactions by structural matching". In: *Nucleic acids research* (2005), W331–W336.

[110] Patrick Aloy and Robert B Russell. "InterPreTS: Protein Inter action Pre diction through T ertiary S tructure". In: *Bioinformatics* (2003), pp. 161–162.

[111] Patrick Aloy and Robert B Russell. "Interrogating protein interaction networks through structural biology". In: *Proceedings of the National Academy of Sciences* (2002), pp. 5896–5901.

[112] Stefan R Maetschke et al. "Gene Ontology-driven inference of protein–protein interactions using inducers". In: *Bioinformatics* (2012), pp. 69–75.

[113] Sanghamitra Bandyopadhyay and Koushik Mallick. "A new feature vector based on gene ontology terms for protein-protein interaction prediction". In: *IEEE/ACM transactions on computational biology and bioinformatics* (2016), pp. 762–770.

[114] Haohan Wang and Madhavi K Ganapathiraju. "Evaluation of Protein-protein Interaction Predictors with Noisy Partially Labeled Data Sets". In: *arXiv preprint arXiv:1509.05742* (2015).

[115] Yanzhi Guo et al. "Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences". In: *Nucleic acids research* (2008), pp. 3025–3030.

[116] Yu Zhen Zhou et al. "Prediction of protein-protein interactions using local description of amino acid sequence". In: *Advances in Computer Science and Education Applications*. Springer, 2011, pp. 254–262.

[117] Lei Yang et al. "Prediction of protein-protein interactions from protein sequence using local descriptors". In: *Protein and Peptide Letters* (2010), pp. 1085–1090.

[118] Juwen Shen et al. "Predicting protein–protein interactions based only on sequences information". In: *Proceedings of the National Academy of Sciences* (2007), pp. 4337–4341.

[119] Zhu-Hong You et al. "Predicting protein-protein interactions from primary protein sequences using a novel multi-scale local feature representation scheme and the random forest". In: *PLoS One* (2015), e0125811.

[120] Zhu-Hong You et al. "Prediction of protein-protein interactions from amino acid sequences using a novel multi-scale continuous and discontinuous feature set". In: *BMC bioinformatics*. BioMed Central. 2014, S9.

[121] Leon Wong et al. "Detection of protein-protein interactions from amino acid sequences using a rotation forest model with a novel pr-lpq descriptor". In: *International Conference on Intelligent Computing*. Springer. 2015, pp. 713–720.

[122] Yu-An Huang et al. "Using weighted sparse representation model combined with discrete cosine transformation to predict protein-protein interactions from protein sequence". In: *BioMed research international* (2015).

[123] Tanlin Sun et al. "Sequence-based prediction of protein protein interaction using a deep-learning algorithm". In: *BMC bioinformatics* (2017), p. 277.

[124] Somaye Hashemifar et al. "Predicting protein–protein interactions through sequence-based deep learning". In: *Bioinformatics* (2018), pp. i802–i810.

[125] Hang Li et al. "Deep Neural Network Based Predictions of Protein Interactions Using Primary Sequences". In: *Molecules* (2018), p. 1923.

[126] Muhao Chen et al. "Multifaceted protein–protein interaction prediction based on Siamese residual RCNN". In: *Bioinformatics* (2019), pp. i305–i314.

[127] Kevin K Yang et al. "Learned protein embeddings for machine learning". In: *Bioinformatics* (2018).

[128] Ariel S Schwartz et al. "Deep semantic protein representation for annotation, discovery, and engineering". In: *BioRxiv* (2018), p. 365965.

[129] Xiaomei Wu et al. "Prediction of yeast protein–protein interaction network: insights from the Gene Ontology and annotations". In: *Nucleic acids research* (2006), pp. 2137–2150.

[130] John P Miller et al. "Large-scale identification of yeast integral membrane protein interactions". In: *Proceedings of the National Academy of Sciences* (2005), pp. 12123–12128.

[131] Philip Resnik. "Using information content to evaluate semantic similarity in a taxonomy". In: *arXiv preprint cmp-lg/9511007* (1995).

[132] Jay J Jiang and David W Conrath. "Semantic similarity based on corpus statistics and lexical taxonomy". In: *arXiv preprint cmp-lg/9709008* (1997).

[133] Dekang Lin et al. "An information-theoretic definition of similarity." In: *Icml*. 1998, pp. 296–304.

[134] Andreas Schlicker et al. "A new measure for functional similarity of gene products based on Gene Ontology". In: *BMC bioinformatics* (2006), p. 302.

[135] Sanghamitra Bandyopadhyay and Koushik Mallick. "A new path based hybrid measure for gene ontology similarity". In: *IEEE/ACM transactions on computational biology and bioinformatics* (2013), pp. 116–127.

[136] Asa Ben-Hur and William Stafford Noble. "Kernel methods for predicting protein–protein interactions". In: *Bioinformatics* 21.suppl_1 (2005), pp. i38–i46.

[137] Oznur Tastan et al. "Prediction of interactions between HIV-1 and human proteins by information integration". In: *Biocomputing 2009*. World Scientific, 2009, pp. 516–527.

[138] Yungki Park and Edward M Marcotte. "Flaws in evaluation schemes for pair-input computational predictions". In: *Nature methods* (2012), p. 1134.

[139] Martin H Schaefer et al. "HIPPIE: Integrating protein interaction networks with experiment based quality scores". In: *PloS one* (2012), e31826.

[140] Leo Breiman. "Random forests". In: *Machine learning* (2001), pp. 5–32.

[141] Jianmin Wu et al. "Integrated network analysis platform for protein-protein interactions". In: *Nature methods* (2009), p. 75.

[142] Ioannis Xenarios et al. "DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions". In: *Nucleic acids research* (2002), pp. 303–305.

[143] Shawn Martin et al. "Predicting protein–protein interactions using signature products". In: *Bioinformatics* (2004), pp. 218–226.

[144] Jean-Philippe Vert et al. "A new pairwise kernel for biological network inference with support vector machines". In: *BMC bioinformatics*. BioMed Central. 2007, S8.

[145] Yijie Ding et al. "Predicting protein-protein interactions via multivariate mutual information of protein sequences". In: *BMC bioinformatics* (2016), p. 398.

[146] Yiwei Li and Lucian Ilie. "SPRINT: ultrafast protein-protein interaction prediction of the entire human interactome". In: *BMC bioinformatics* (2017), p. 485.

[147] Sea Pitre et al. "Global investigation of protein–protein interactions in yeast Saccharomyces cerevisiae using re-occurring short polypeptide sequences". In: *Nucleic acids research* (2008), pp. 4286–4294.

[148] Matteo Bellucci et al. "Predicting protein associations with long noncoding RNAs". In: *Nature methods* (2011), p. 444.

[149] Zhu-Hong You et al. "Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis". In: *BMC bioinformatics*. BioMed Central. 2013, S10.

[150] Xiuquan Du et al. "DeepPPI: boosting prediction of protein–protein interactions with deep neural networks". In: *Journal of chemical information and modeling* (2017), pp. 1499–1510.

[151] Paul Shannon et al. "Cytoscape: a software environment for integrated models of biomolecular interaction networks". In: *Genome research* (2003), pp. 2498–2504.

[152] Damian Szklarczyk et al. "The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible". In: *Nucleic acids research* (2016), gkw937.

[153] Dhananjay Kimothi et al. "Distributed Representations for Biological Sequence Analysis". In: *arXiv preprint arXiv:1608.05949* (2016).