ENZYME PREDICTION WITH WORD EMBEDDING APPROACH

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ERKAN AKIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

ENZYME PREDICTION WITH WORD EMBEDDING APPROACH

submitted by **ERKAN AKIN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. M. Volkan Atalay
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering, METU _____

Prof. Dr. M. Volkan Atalay
Computer Engineering, METU _____

Assoc. Prof. Dr. Tunca Doğan
Institute of Informatics, Hacettepe University _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Erkan Akın

Signature :

ABSTRACT

ENZYME PREDICTION WITH WORD EMBEDDING APPROACH

Akın, Erkan
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. M. Volkan Atalay

September 2019, 78 pages

Information such as molecular function, biological process, and cellular localization can be inferred from the protein sequence. However, protein sequences vary in length. Therefore, the sequence itself cannot be used directly as a feature vector for pattern recognition and machine learning algorithms since these algorithms require fixed length feature vectors. We describe an approach based on the use of the Word2vec model, more specifically continuous skip-gram model to generate the vector representation of a given protein sequence. In the Word2vec model, a protein sequence is treated as a document or a sentence and its subsequences correspond to words. The continuous skip-gram model is a supervised Word2vec model to predict the surrounding subsequences from a subsequence. Feature vectors from the Word2vec model can be coupled with classifiers to infer information from the sequence. As a sample application, we consider the problem of determining whether a given protein sequence is an enzyme or not. For a sample dataset that contains 19,155 of enzyme and non-enzyme protein sequences, for which 20% of these sequences are put apart for test and 80% is used for 5-fold cross-validation. The best performance scores are obtained as 0.97 for Precision, Recall, F1, accuracy and 0.93 for Matthews correlation

coefficient by the Word2vec model with vector size of 100, the window size of 25 and number of epochs as 180 and for the Random Forest classifier. Also, we generate vector representations for the first level of Enzyme Commission classes by using the same hyper-parameter set for the Word2vec model. For vector representations of each class, binary classification is applied and the average performance scores are obtained as 0.87 for Precision, Recall, F1, accuracy and 0.70 for Matthews correlation coefficient by using the Random Forest classifier.

Keywords: Word2vec, Word Embedding, Proteins, Classification, Enzymes

ÖZ

KELİME YERLEŞTİRME YAKLAŞIMI İLE ENZİM TAHMİNİ

Akın, Erkan
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. M. Volkan Atalay

Eylül 2019 , 78 sayfa

Moleküler fonksiyon, biyolojik işlem ve hücresel lokalizasyon gibi bilgiler protein sekansından çıkarılabilir. Bununla birlikte, protein sekanslarının uzunluğu değişir. Bu nedenle, desen tanıma ve makine öğrenme algoritmaları sabit uzunluklu özellik vektörleri gerektirdiğinden, sekans bu algoritmalar için doğrudan bir özellik vektörü olarak kullanılamaz. Belirli bir protein sekansının vektör gösterimini oluşturmak için Word2vec modelinin, daha spesifik olarak sürekli atlamalı modelin kullanımına dayanan bir yaklaşımı tarif ediyoruz. Word2vec modelinde, bir protein sekansı belge ya da cümle olarak ele alınır ve onun alt sekansları kelimelere karşılık gelir. Sürekli atlama modeli, bir alt sekansı çevreleyen alt sekansları tahmin etmek için kullanılan bir Word2vec modelidir. Word2vec modelindeki özellik vektörleri, sekanstan bilgi almak için sınıflandırıcılarla birleştirilebilir. Örnek bir uygulama için, bir protein sekansının enzim olup olmadığını belirleme sorununu ele alıyoruz. 19,155 enzim ve enzim olmayan protein sekansı içeren örnek bir veri seti için, bu dizilerin 20%'si test için ayrılmış ve 80%' i 5'li çapraz validasyon için kullanılmıştır. En iyi performans sonuçları, Word2vec modelinin parametreleri için 100 vektör büyüklüğü, 25 pencere boyutu ve 180 tekrarlama sayısı kullanılmış ve Rassal Orman sınıflandırıcısı için Pre-

cision, Recall, F1 ve Doğruluk sonuçları 0.93 ve Matthews korelasyon katsayısı 0.97 olarak bulunmuştur. Ayrıca, Word2vec modeli için kullanılan aynı parametreleri kullanarak ilk seviye Enzim Komisyonu sınıfları için vektör gösterimleri üretiyoruz. Her bir sınıfın vektör gösterimleri için ikili sınıflandırma uygulanır ve Rassas Orman sınıflandırıcısını kullanarak ortalama performans Matthews korelasyon katsayısı için 0.70 ve Precision, Recall, F1 ve Doğruluk sonuçları için 0.86 olarak elde edilir.

Anahtar Kelimeler: Word2vec, Word Yerleştirme, Proteinler, Sınıflandırma, Enzimler

To my family

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Mehmet Volkan Atalay for the continuous support, his patience, motivation, and immense knowledge. His guidance helped me during this study and writing of this thesis. Also, I would like to thank Ahmet Rifaioğlu for his guidance about the bioinformatic studies and their applications.

I would like to express my gratitude to my colleagues for their support throughout the writing of this thesis. It was very valuable to me to get their support. It is always nice to work with them.

Last but not least, I would like to thank my family for continuous support. Thanks to my father, Erhan Akın, for the motivation he gave me. I also would like to thank my mother, Sema Akın, and my brother, Eray Akın, for their support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

EC	Enzyme Commission
KNN	K-Nearest Neighbor
RF	Random Forest
AdaBoost	Adaptive Boosting
SVM	Support Vector Machine
ANN	Artificial neural network
RBF	Radial Basis Function
CBOW	Continuous Bag-Of-Words
MCC	Matthews correlation coefficient
PCC	Pearson correlation coefficient
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
SNE	Stochastic Neighbor Embedding
MCL	Markov Clustering Algorithm
TransClust	Transitive Clustering
SMILES	Simplified Molecular Input Line Entry System
HMM	Hidden Markov model
PWM	Position weighted matrix
DM	Distributed-memory
DBOW	Distributed Bag of Words
BLAST	Basic Local Alignment Search Tool
FASTA	Fast All

UniProt The Universal Protein Resource

CHAPTER 1

INTRODUCTION

Enzymes are biological catalysts which speed up the biological reactions which are essential processes for organisms. Enzymes are proteins which are used to lower the activation energy for biological reactions. Recent progress in technology and studies show that the number of functionally uncharacterized proteins is increasing. Representation of biological structures such as sequences has advantages for applying machine learning applications. The main objective of this study is the prediction of protein functions from protein sequences by machine learning methods. There are several essential protein functions such as biochemical reactions, creating tissue structures, maintaining communication between cells, tissues, and organs, protecting the body from invaders and providing energy when the body needs. Since proteins have many functions, they need to be classified. Gene Ontology Annotation and Enzyme Commission Nomenclature are used to represent the functions of a protein sequence. A GO annotation is a vocabulary that represents the function of a gene or a protein [1]. Besides, protein can be categorized as an enzyme or non-enzyme. In order to classify enzymes, Enzyme Commission numbers are defined. Enzyme Commission (EC) Nomenclature is a classification scheme based on the reaction in which the enzyme catalyzes.

An enzyme is a protein and enzymes can be represented by an amino acid sequence which holds information to identify protein functions [2]. One of the alternatives to predict the functions of proteins from the sequence is to apply natural language processing techniques on the protein sequences. Since protein sequences hold information about proteins, the function of proteins can be referred from sequences and based on the relations among the subsequences. If the function of protein is catalyz-

ing chemical reactions, protein can be classified as enzyme and associated with an EC class.

1.1 Problem Statement

The first problem is the characterization process of proteins and wet laboratory experiments on proteins are time-consuming and expensive, alternative approaches such as the use of computational prediction methods became more important. The second problem is that proteins have a varying length so protein sequences cannot be used directly for prediction models. Machine learning algorithms can be applied to the biological structures to predict the function. These machine learning algorithms include algorithms for classification and natural language processing. The application of natural language processing techniques to biological sequences is the solution method that is offered by this study.

Another problem is tracing the relations between predicted enzymes when too many protein sequences are given to prediction models. Enzyme Commission Nomenclature defines the relations of enzymes according to their functions. The relationship of enzymes are represented with the names of the enzyme classes which use abbreviations that give the base class and the name of the class. In order to trace these relationships, individual relationships should be followed. Visual representation of the Enzyme Commission classes is a way to represent the relations between protein sequences.

1.2 Approach

Word embedding is referred to as continuous vector representation which is the term of mapping a set of words or phrases to a continuous vector space. In natural language processing, continuous vector representation is an efficient way to represent words of large datasets. A continuous vector representation technique, Word2vec, is first purposed to construct word vectors from a large dataset [3]. Biological sequences can be split into consecutive amino acid groups which are subsequences. For this

study, the subsequences are treated as words in a sentence. Also, the sequences are used as sentences. The k-mer refers to consecutive subsequences of amino acids where the length, k, is referred to as l in the rest of the manuscript. Each subsequence of the protein sequence can be embedded in a vector and represented as an index in a subsequence vocabulary. The vocabulary of subsequences is constructed from a given training dataset. These subsequences map to vectors of numerical values. The vector representation of a protein sequence is computed by using the vector representations of the subsequences of the protein sequence.

Sequences of functionally annotated and manually reviewed proteins are taken from the SwissProt section of The Universal Protein Resource Knowledgebase (UniProtKB) database. Word2vec results in a vector for each subsequence and the vectors are further used as feature vectors for input to the machine learning algorithms. In this thesis, machine learning algorithms are mainly classifiers such as k-Nearest Neighbor (kNN), Support Vector Machine (SVM), Naive Bias, Random Forest classifier and AdaBoost classifier. Feature vector values of labeled sequences and their corresponding class values are used for training the binary classifiers which are then employed to predict the class values of unlabeled sequences. For a protein, there should be a single feature vector that is constructed from its subsequence vectors. There are three alternatives to construct sequence feature vectors. The first one is getting maximum values of columns of the subsequence feature vectors. The second alternative is getting minimum values as similar to maximum value extraction. The last alternative is calculating the summation of subsequence feature vectors and normalizing them.

There are two alternatives to adjust the hyper-parameters of the Word2vec model correctly. The first one is comparing word vector differences (similarities) with sequence homology of protein sequences which can be calculated by BLAST [4]. The second way is based on the accuracy results of classifiers. Both of these alternatives are used in this study.

Visualization of word embedding vectors is a key step before classifier selection. If the feature vectors are projected to form groups of data points in Euclidean space, the same vectors should be separable. There are a few methods for dimensional reduction which embed high dimensional data in two-dimensions. One of these methods

is t-distributed stochastic neighbor embedding (t-SNE) which is a machine-learning algorithm to project high dimensional data onto two dimensional space. Word2vec model uses high dimensional vector representations and these vector representations are projected onto the two-dimensional space by using the t-SNE algorithm. Another step is choosing a successful classifier. In order to choose a successful classifier, the values of hyper-parameters of several alternative classifiers are determined and the performance of these classifiers are assessed.

The last step is visualizing the results of predictions of biological sequences. The relations between predicted enzymes are difficult to track and analyze since there can be too many predictions. Visualizing the EC numbers as a tree is an approach to show the relations between predicted enzymes. Also, the distribution of the number of the enzymes in classes is important to show which class and subclasses of that class are predicted mostly. EC numbers and their associated enzymes are visualized in a web-based tool by using graph visualization techniques and a coloring mechanism is implemented to show the distribution of prediction results.

1.3 Improvements

This thesis describes two specific improvements which are given as follows:

- Word2vec model is applied to protein sequences to predict the functions of proteins such as enzyme classes of proteins.
- A web-based interactive visualization tool is developed to visualize the predictions of Enzyme Commission numbers.

The word embedding approach, Word2vec, is applied on biological sequences and the target dataset contains enzyme classes of proteins. We measure the performance scores for different lengths of subsequences such as 3 and 5. The previous studies [5] [6] use 3 as the length of subsequences of protein sequences. In this study, the protein sequences are divided into subsequences which have 5 amino acids (5-mer) for each subsequence to achieve better classification performance. Also, we use several classifiers to predict unlabeled datasets and compare the performances of these

classifiers. The predicted Enzyme Commission classes are visualized to show the relations between classified enzyme sequences.

CHAPTER 2

BACKGROUND INFORMATION AND RELATED WORK

2.1 Amino Acids

Amino acids are organic molecules that contain carbon (C), hydrogen (H), oxygen (O) and nitrogen (N). Other elements such as Sulfur (S) could be found in certain amino acids. Amino acids are represented by an alphabet of 20 symbols. These symbols are A, C, D, E, F, G, H, I, J, K, L, M, N, P, Q, R, S, T, V, W, and Y. The amino acids synthesize proteins. A protein sequence, w, is a string generated with this alphabet. The length of the protein sequence can be represented as L which is the number of symbols in the string. In addition to protein sequences, there are other biological sequences such as nucleic acid sequences (DNA, RNA). For nucleic acid sequences, the alphabet is composed of four letters (A, C, G, and T).

2.2 Proteins

Proteins are molecules that play crucial roles in biological organisms. Proteins are required for energy generation, protection of the body, cell structure and communication between cells. These functions include catalyzing biochemical reactions and DNA replication. Amino acids are the components of proteins and amino acids bind together to form proteins. There are 20 types of amino acids and different combinations of amino acids form different proteins. Similar to the other molecules, proteins are three-dimensional structures and these structures are unique for each protein. The amino acids are bond with peptide bonds. A small number (20 to 30) of amino acids forms peptides. A linear chain of amino acids is a polypeptide. At least one of long

polypeptides is contained by proteins. The sequence of the proteins are determined by the DNA sequence of a gene. Proteins can interact with other molecules to catalyze reactions or replicate DNA.

The structure of proteins can be represented with different structure types such as primary structure, secondary structure, and tertiary structure. The tertiary structure is the shape of a protein molecule and contains associated secondary structures. The secondary structure composed of repeating structures in proteins. The primary structure is an amino acid sequence in one dimension. In Figure 2.1, 3D and 1D structures of one protein, TRY2_RAT, are shown for illustration purposes. Obtaining the 3D structure of a protein requires laboratory experiments. Since the primary structures are relatively easy to obtain, public databases such as The Universal Protein Resource (UniProt)/SwissProt hold the one-dimensional protein sequences. Protein sequencing is a technique for determining the amino acid sequence of a protein.

>sp|P00763|TRY2_RAT Anionic trypsin-2 OS=Rattus norvegicus OX=10116 GN=Prss2 PE=1 SV=2
MRALLFLALVGAAVAFPVDDDDKIVGGYTCQENSVPYQVSLNSGYHFCGGSLINDQWVVS
AAHCYKSRIQVRLGEHNINVLEGNEQFVNAAKIIKHPNFDRKTLNNDIMLIKLSSPVKLN
ARVATVALPSSCAPAGTQCLISGWGNTLSSGVNEPDLLQCLDAPLLPQADCEASYPGKIT
DNMVCVGFLEGGKDSCQGDSGGPVVCNGELQGIVSWGYGCALPDNPGVYTKVCNYVDWIQ
DTIAAN

Figure 2.1: 3D Structure and sequence of TRY2_RAT.

2.3 Gene

Genes are structures that provide physical inheritance in living organisms. DNA is the basic building block of the gene. Gene Ontology (GO) is an ontology which is used to represent functions of genes and gene products from organisms present the related information in a machine-readable form. Wet laboratory experiments about genes and gene products provide results about the functional information of gene products. GO annotation includes references such as journal articles and gene product identifiers (protein ids and GO terms).

2.4 Enzymes

Most of the enzymes are proteins that catalyze biochemical reactions. In order to accelerate reactions, enzymes lower the activation energy which is required to complete reactions. Some of the enzymes aren't proteins and these enzymes are RNA molecules. The enzymes act with molecules and they are called as substrates which can be defined as products. Enzymes catalyze the metabolic processes in cells in order to speed up reactions to sustain life.

For some reactions, substrates are broken into multiple products and some reactions create a larger product. In Figure 2.2, the substrate is broken into products with the help of an enzyme.

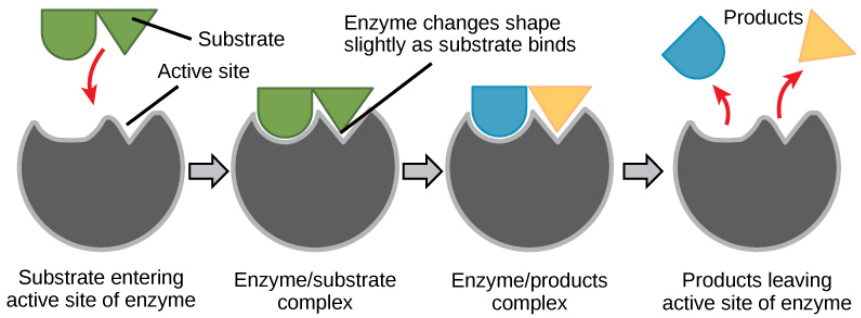

Figure 2.2: Reactions with enzymes reproduced from [7].

Classification of enzymes is defined as a scheme which is called Enzyme Commission

(EC) Nomenclature. EC number is a four-digit representation and digits represent the function of enzyme general to a specific order. Four levels of EC numbers are related to each other in a functional hierarchy. Within the first level, the system annotates the main enzymatic classes (i.e., 1: oxidoreductases, 2: transferases, 3: hydrolases, 4: lyases, 5: isomerases and 6: ligases).

2.5 Sequence Alignment

Sequence alignment is a method to identify similarities between biological sequences such as DNA, RNA or protein sequences. These similarities help scientists to observe functional, or structural relations between biological sequences [8]. Aligned subsequences could correspond to functional regions, but a single region of the biological sequence isn't enough to associate functions [9]. The alignment of sequences requires computational methods since the lengths of sequences vary. One of the computational methods is dynamic programming which is the common method for sequence alignment. Global and local alignments are two ways of aligning sequences. Needleman–Wunsch [10] algorithm is a global alignment method that uses dynamic programming as a fundamental computation method. For local alignment, Smith–Waterman [11] algorithm is a common method that uses dynamic programming similar to the Needleman–Wunsch algorithm.

2.6 Proposed Methods and Models

Word2vec has a hidden layer that holds real-valued feature vectors which provide information about similarities between the given biological sequences. Also, BLAST finds similarities between sequences and these similarities are compared with the cosine similarities of the feature vectors which are obtained from Word2vec. Flowchart of the approach of this study is given in Figure 2.3. Hyper-parameters of the Word2vec model is optimized by using similarities of the protein sequences. Enzyme Commission classes are predicted by training classifiers with feature vectors of protein sequences. The t-SNE algorithm is used to project feature vectors onto the two-dimensional space.

Classification includes *k*-Nearest Neighbors, Support Vector Machine, Naive Bayes, Random Forest, AdaBoost, and a shallow feed-forward neural network.

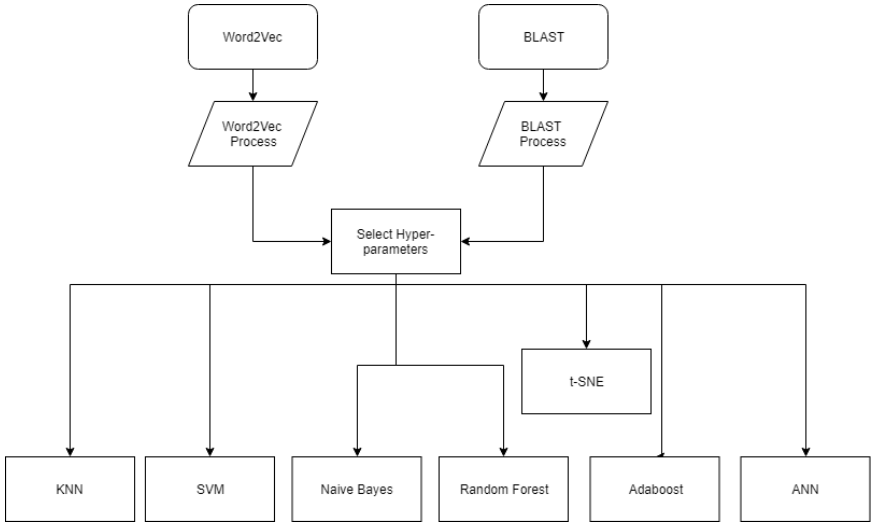

Figure 2.3: Flowchart of the proposed methods where the Word2vec model is trained with subsequences and feature vectors of these subsequences are fetched from the Word2vec model to compute feature vectors of protein sequences. With these feature vectors, classifiers are trained to predict Enzyme Commission classes.

2.6.1 Word2vec Model

Word2vec is a shallow neural network which is employed to produce feature vectors from a given text corpus. A vocabulary is constructed from the given training text corpus which contains the words from the manually validated sentences. Word2vec model then creates a vector representation of words from the given vocabulary. The resulting word vector can be used as a feature vector for machine learning applications.

There are two models of Word2vec. The first one is continuous bag-of-words (CBOW) and the second one is skip-gram. Continuous bag-of-words model predicts the current word from its surrounding words around the current word. A window (W) is defined to include the number of surrounding words. Figure 2.4 shows the diagram for CBOW model. Future words ($w_{c+1}..., w_{c+W}$) and history ($w_{c-W}, ..., w_{c-1}$) words are given to CBOW model as inputs. The CBOW model predicts the current word (w_c)

from the given future and history words. Figure 2.5 show diagram for Skip-gram. Skip-gram predicts future words and history words ($w_{c-W}, ..., w_{c-1}, w_{c+1}..., w_{c+W}$) based on current word (w_c). Skip-gram model uses the current word to predict its surrounding words in a defined window.

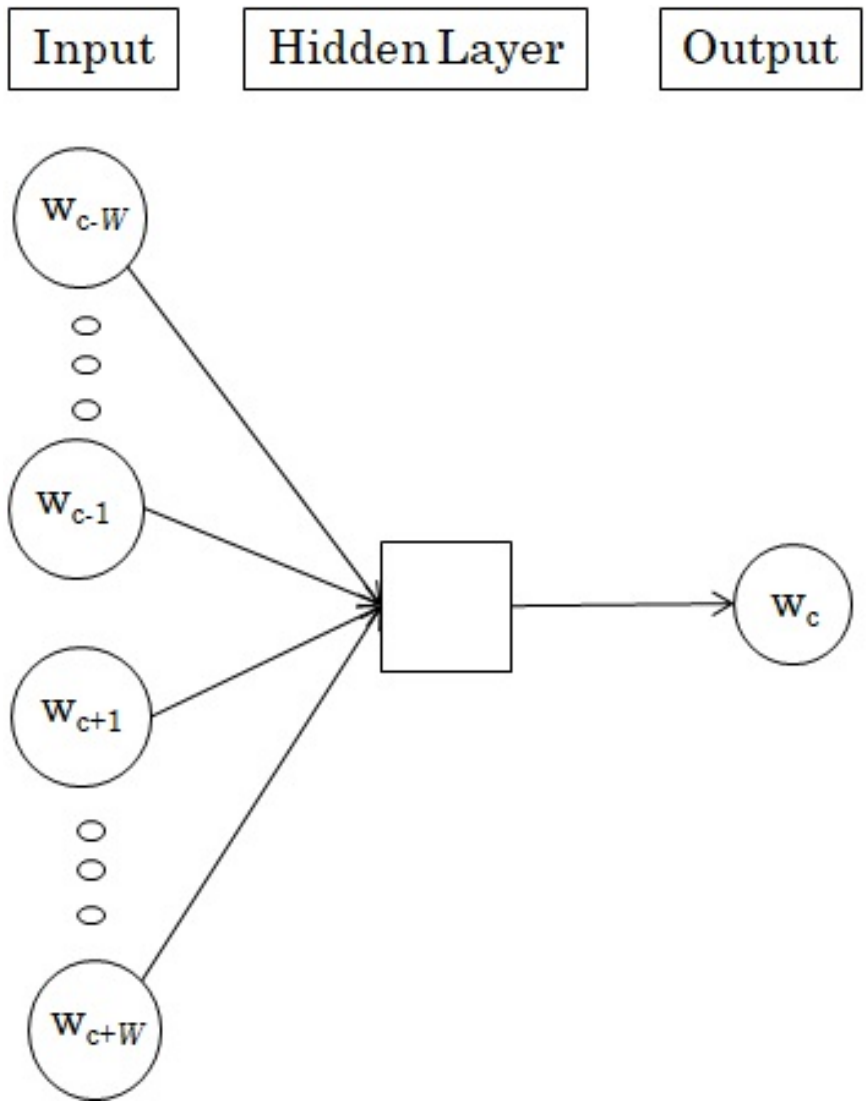

Figure 2.4: Word2Vec Continuous bag-of-words model.

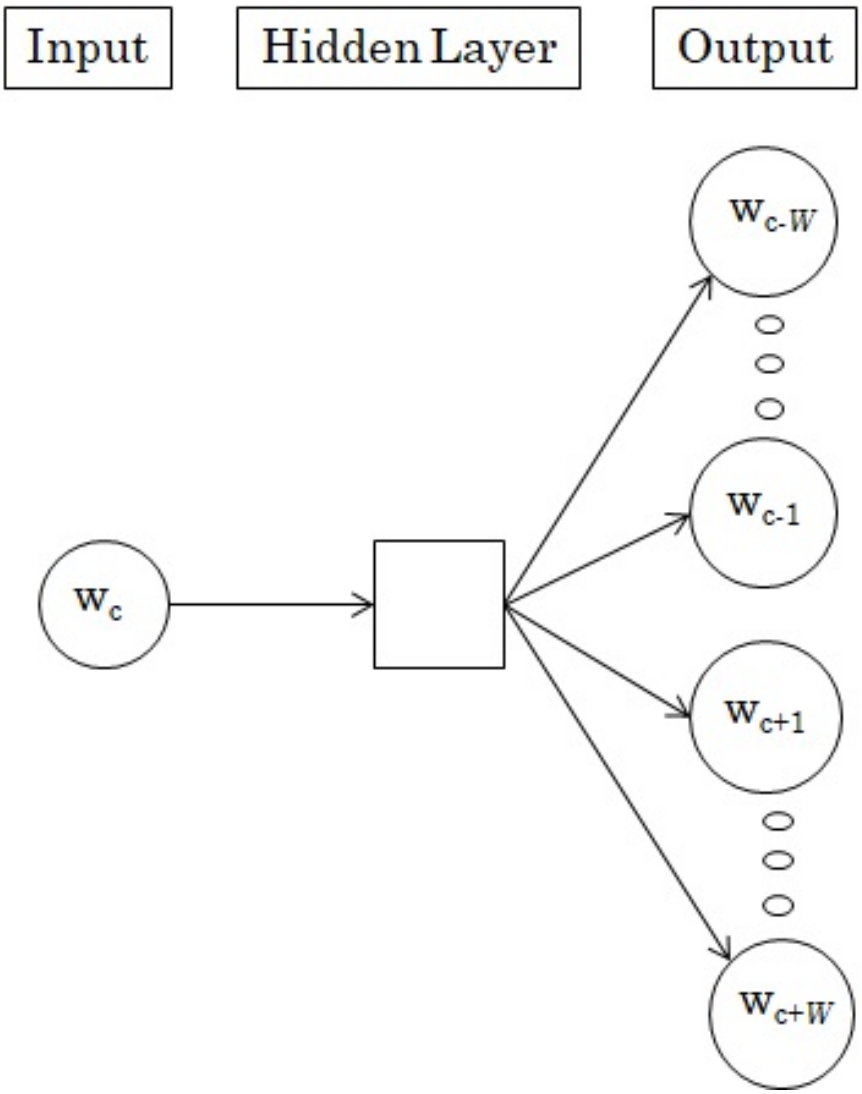

Figure 2.5: Word2Vec Skip-gram model.

Word2vec training depends on several hyper-parameters including vector size (N), window (W), minimum frequency and number of epochs. The number of epochs is similar to the one in other machine learning methods. Epochs indicate the number of iterations that Word2vec model works through the training dataset. Minimum frequency indicates the minimum number of occurrences of a word to be included by Word2vec model training. A word that has a lower frequency than the minimum frequency parameter is ignored and the vocabulary of Word2vec model doesn't contain the word. The window size (W) parameter is the maximum number of words between the current word and the surrounding words. Vector size (N) determines the number

of nodes in the hidden layers of the Word2vec model.

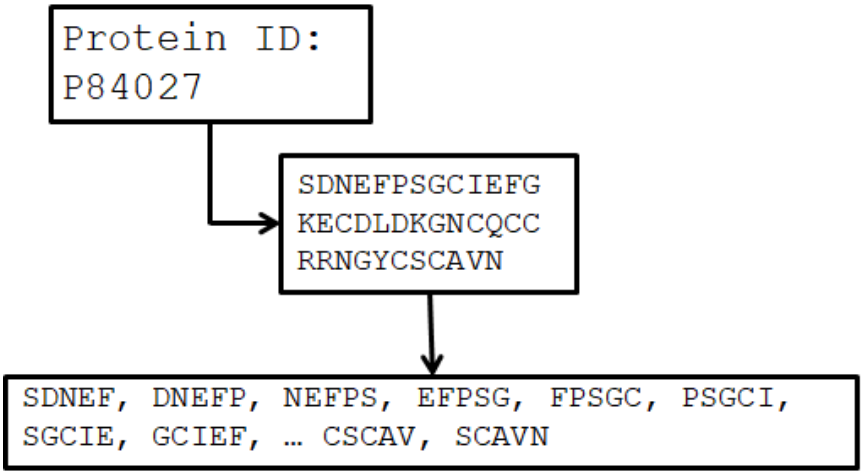

Figure 2.6: Overlapped 5-mers subsequences of P84027.

Word2vec model is trained with given subsequences of protein sequences. Figure 2.6 illustrates the protein sequence of a protein, P84027, and overlapping 5-mers subsequences of the sequence. Word2vec represents subsequences as a vector space model which is an algebraic model for representing subsequences as a vector of numerical values.

2.6.2 BLAST

Basic Local Alignment Search Tool (BLAST) is a tool that finds similarities between biological sequences with local sequence alignment algorithm which is described in Section 2.5. BLAST compares a biological sequence to another biological sequence and the result of the comparison can be used to identify functional similarities. The E-Value is a threshold value that measures the statistical significance of the similarity between biological sequences. The match between a region of sequences could occur by chance. The probability of this chance is defined as E-value. The default E-value is 10 and higher E-values indicate that BLAST can report sequences with lower similarities.

2.6.3 *k*-Nearest Neighbors

The k-nearest neighbor (kNN) is a classifier that assigns unlabeled data with respect to the nearest labeled set of data. Nearest labeled set of data consists of k data items which are closest k data items from the labeled set of data. These k data items are referenced as k nearest neighbors. *K*NN classifier is one of the simplest classifiers.

Algorithm 1 is the pseudocode of *k*-Nearest Neighbors which is adapted from Tay, Hyun, and Oh [12]. The classification phase includes majority voting among the data items in the neighborhood. Distance values between all of the pairs are calculated and stored. If an unlabeled data is going to be classified, the nearest k data items are chosen. The distance between data items can be defined as Euclidean or Manhattan or Mahalanobis distance. Let the number of features be n and x and y be two data items. The Euclidean distance (d) is defined as given in Equation(2.6.1).

For kNN, it is important to choose the best value for k in order to classify training data items correctly. The grid search approach is useful to choose k value which is optimized by using the performance metrics of kNN evaluation results which are collected by cross-validation on the training set.

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + .. + (x_n - y_n)^2} \tag{2.6.1}$$

Algorithm 1 Pseudocode of *k*-Nearest Neighbors [12]

Require: X: set of labeled data, Y: class labels of X, x: unknown sample, y: labeled data, m: number of data items in the training dataset

for $i \leftarrow 1$ to m **do**

$y \leftarrow X_i$

compute distance between x and y according to Equation 2.6.1

end for

compute the set, I, which contains the indices for the k smallest distances between $X_1 \ldots X_m$ and x

return majority label for Y_i where $\{i$ is in $I\}$

2.6.4 Naive Bayes

Naive Bayes is a probabilistic classifier that uses Bayes' Theorem to classify data. Probabilistic classifiers use feature vectors and classes to determine the probability of the data items belonging to each class. For each feature vector, the most likely class is computed based on Bayes' theorem with the assumption of independence of features. Since binary classification is applied to protein function prediction, positive and negative classes can be named as c_0 and c_1. Let the feature vectors be v_1, v_2 .. v_n and P be the symbol of probability. The representation of naive Bayes for a class (c_0) probability is given in Equation (2.6.2) which shows the conditional distribution of feature vector over class c_0. $P(c_0)$ is the probability of class c_0 which can be calculated as dividing the number of data items in c_0 by the total number of data items. The assumption concerning the independence of the variables may be incorrect but Naive Bayes is a useful and simple classifier.

$$P(c_0|v_1, ..., v_n) = P(c_0) \prod_{i=1}^{n} P(v_i|c_0) \tag{2.6.2}$$

The probability distribution of class c_0 can be computed under the Gaussian distribution which is given by Equation (2.6.3). For Gaussian distribution, the mean and variance of feature vectors (v_1, v_2 .. v_n) are computed. The mean value of class values is represented by μ_{c_0} and variance value is represented with σ_{c_0}. Naive Bayes has the advantage of a short training time. Before the prediction of test data, only parameters of probability distributions are computed. Classification with Naive Bayes is simple. Given the feature vector, the conditional probability of feature vector is computed and class with the highest probability is predicted as a result.

$$P(v_i|c_0) = \frac{1}{\sqrt{2\pi\sigma_{c_0}^2}} e^{(-\frac{(v_i-\mu_{c_0})^2}{2\sigma_{c_0}^2})} \tag{2.6.3}$$

Algorithm 2 Pseudocode of Naive Bayes Classifier [13]

Require: T: training dataset, C: The value of the predictor variable in testing dataset.

C=(c_0, c_1)

Read the training dataset T

Calculate μ and σ of the predictor variables in each class

for $i \leftarrow 0$ to 1 **do**

Calculate the probability of c_i using the Gaussian distribution which is given in Equation (2.6.3)

end for

Calculate the likelihood for each class which is given in Equation (2.6.2)

return Greatest likelihood

2.6.5 Support Vector Machine

Support Vector Machine (SVM) is a classifier that is used for both regression and classification tasks. SVM is a supervised learning model that uses the given data to construct a classifier model. The goal of SVM is finding a hyperplane in N-dimensional space where N indicates the number of features. In order to apply binary classification, there can be many hyperplanes but optimal hyperplane should be selected. The optimal hyperplane should maximize the margin distance between data items of the two classes.

There are many methods for constructing hyperplane. The first method is the linear classifier. The second method is non-linear classification which maps inputs into higher dimensional space. The maximum margin hyperplane divides data items into groups of data items. Figure 2.7 shows data items that are separated by a hyperplane in 2D and 3D space. In some cases, data items can not be separated linearly which requires an optimized the hyperplane.

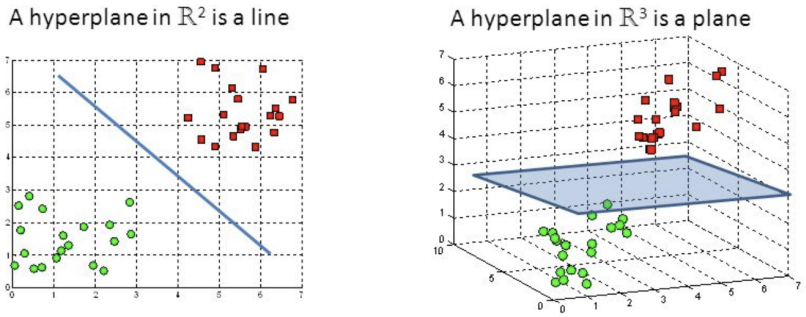

Figure 2.7: Two sets of data items are separated by hyperplanes in 2D and 3D space. Image is reproduced from [14].

In 1992, Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik propose a new way to construct hyperplanes which are non-linear [15]. They applied kernel trick [16] for maximum margin hyperplanes. Non-linear classifiers use kernel functions to project data items onto higher dimensional space in which a hyperplane can be fit to separate data items. Also, non-linear hyperplanes give an advantage of representing feature vectors in a higher dimensional space by having more attributes. Although non-linear hyperplanes increase the performance of the SVM classifier, there is a disadvantage which is referred to as the overfitting problem in machine learning algorithms. There are a few numbers of kernel functions such as Polynomial, Radial Basis Function, and Hyperbolic tangent. We focus on one of the kernel functions, Radial Basis Function (RBF), which is used by the SVM classifier.

SVM algorithm has hyper-parameters for Radial Basis Function (RBF) such as γ value and C value. Equation 2.6.4 shows the kernel function for RBF by using a free parameter, γ, where v_i and v_j are the feature vectors. γ parameter defines the range of influence of a training sample. Low γ values indicate far influence. C parameter defines the tolerance value between the hyperplane's margin and the correct classification of training data. If the value of C is larger, the margin is smaller and the decision function is better at classifying training data items correctly which might lead to the overfitting problem. For smaller C values, margin gets larger, therefore the hyperplane misclassifies more data items.

$$K(v_i, v_j) = \exp(-\gamma||v_i - v_j||^2) \tag{2.6.4}$$

2.6.6 Random Forest Classifier

Random forest (RF) is a method for classification of data by constructing several decision trees [17]. From decision trees, predictions are collected and merged into a single decision. The main decision can be made by two main methods. The first method is finding the class which appears most often. The second method is mean prediction (regression) from trees. In order to build a single decision from decision trees, the bagging method is used to create a combination of learning models (decision trees) in order to increase accuracy. Figure 2.8 shows decision trees and how to combine them into a single decision. The decision trees are grown in a random fashion. A random subset of features and training data items are selected. A decision tree is fit to each training sample and the random subset of features is used to build the decision tree. Then, the predicted classes are combined to create a result which is calculated by majority voting. The normal decision trees tend to overfit over-training sets and Random Forest proposes a way to avoid overfitting problem with creating random decision trees.

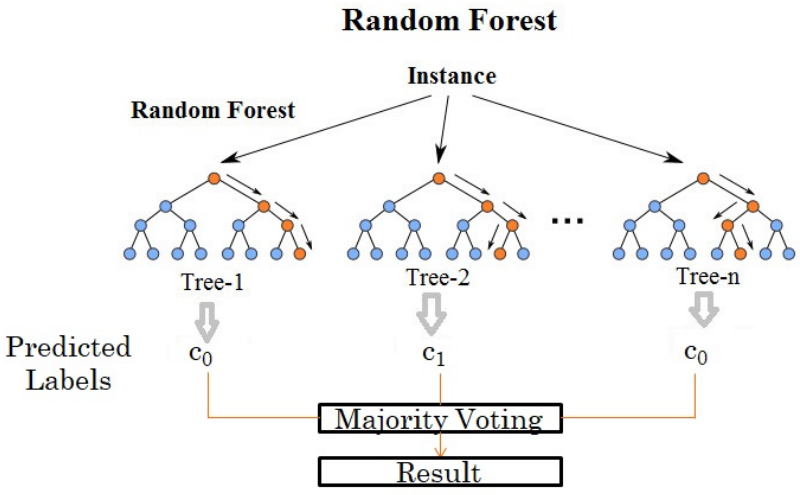

Figure 2.8: Decision Tree image is reproduced from [18]. Data item instance is given to Random Forest Classifier and decision trees are created. The orange nodes show that the decisions which are selected by the decision trees. The main decision node which outputs the final result of Random Forest Classifier by using majority voting.

2.6.7 AdaBoost Classifier

AdaBoost is a short name of Adaptive Boosting. AdaBoost is a meta-estimator which combines the output of the classifiers [19]. AdaBoost makes calls to a weak learning algorithm repeatedly. AdaBoost aims to combine several classifiers that have the assigned weight values to form a better classifier. The weak classifiers can be decision trees that are similar to the Random Forest. At the beginning of training, the weights are set to equal values for each weak classifier. Then, each weak classifier inputs a random subset of labeled data. The accuracy values of weak classifiers change the weight values of these classifiers. The more accurate weak classifier has a higher weight value. After classification of the first iteration, selection method of the random subset of training set changes. Each data item in the training set is assigned a weight value. For misclassified data items from the previous iteration, AdaBoost increases weight values in order to prioritize the data items. As a result, the error of the weak classifiers can be measured and can be minimized by choosing a better weak learner.

2.6.8 Artificial Neural Network

Artificial neural network (ANN) is a computation model that is inspired by the biological neural network. ANN can be used for classification. ANN contains artificial neurons that are similar to biological neurons. Neurons can be referred to as nodes. The model of ANN is made up of an input layer, a hidden layer, and an output layer. These layers consist of several nodes according to number of input and output parameters. Like other classifiers, there are feature vectors and classes of the input data. Input nodes represent feature vectors and output nodes represent classes. The connection between input nodes and output nodes includes hidden layers that hold the information about the relation between inputs and outputs. A simple ANN is shown in Figure 2.9 which has one hidden layer with four nodes. Nodes and edges have weight values that are adjusted during the training process of the neural network. During the classification of test data, feature vectors are fed into the input layer and values of nodes at the output layer are computed according to the weights of edges.

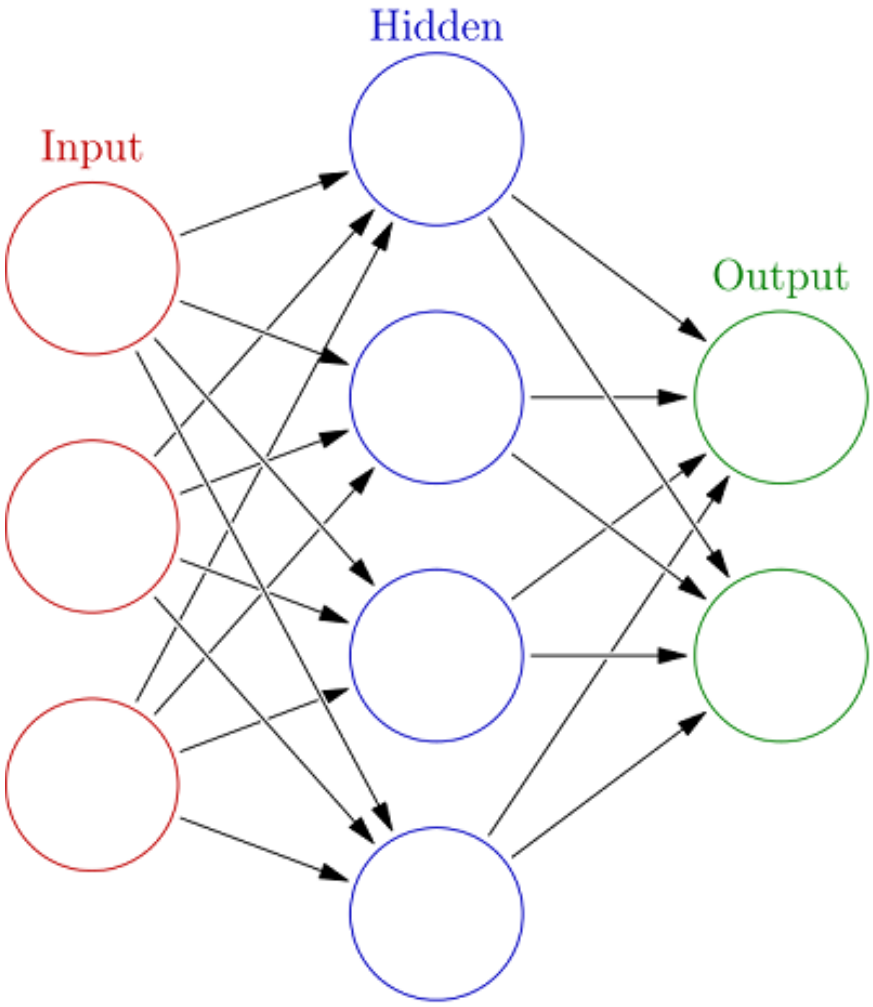

Figure 2.9: Artificial neural network figure is taken from [20]. Since binary classification is applied in this study, two nodes are used at the output layer to illustrate ANN.

The most common method for training is backpropagation which efficiently trains multi-layer networks. Backpropagation is a way to calculation of gradient with the error of outputs [21]. Backward propagation of errors helps the adjustments of the weights. In order to adjust the output of the network, error value should be minimized which can be done with calculating the derivative of the error function. The actual output of a neuron is calculated with the outputs and weights of previous neurons and an activation function such as logistic function.

2.6.9 Measuring the Performance of Classifier

The results of classifiers should be assessed and compared. There are several methods to measure the accuracy of binary classifier models. The accuracy of the Word2vec model and classifiers depend on the set of hyper-parameters such as vector size (N) of the Word2vec model or k value of k-Nearest Neighbors classifier or construction method of hyperplane for SVM. The straightforward method is comparing the true predictions against all predictions. In binary classification, the terms predicted positive and predicted negative refer to the classifier's expected results, and the terms true and false refer to whether that prediction corresponds to the real outcomes of prediction. There are two types of measurements that are used for correctly identified predictions, one is the true positive (TP) and the second one is the true negative (TN). The true positive predictions indicate the proportion of actual positives that are correctly identified. The true negative is the proportion of actual negatives that are correctly identified. Also, incorrectly identified predictions have two types of measurements, one is the false positive (FP) and the second one is the false negative (FN). The false positive indicates that the proportion of actual negatives that are incorrectly identified as positive. The false negative is the proportion of actual positives that are incorrectly identified as negative. Table 2.1 shows the confusion matrix about the relationships between TP, FP, TN and TP and contains the combinations of predicted and actual values. Equation (2.6.5) shows how to calculate accuracy value. The sum of TP and TN values indicates the number of correctly predicted outcomes and the correctly predicted outcomes are divided into the number of all predictions to calculate accuracy.

Table 2.1: Confusion Matrix

Predicted Values	**Actual Positive**	**Actual Negative**
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

In information retrieval, the result of relevancy is measured by precision. Truly relevant results of classifiers are measured by the recall. The high precision value indi-

cates low false positive predictions and high recall value indicates low false negative predictions. Higher precision and recall values show that classifier results have higher accuracy. Equation (2.6.6) shows how to calculate the precision value and Equation (2.6.7) shows how to calculate the recall value. F1 score is a method to calculate the weighted average of precision and recall and Equation (2.6.8) shows how to calculate F1 score by using precision value and recall value. The value of F1 is in range of 0 to 1 and a higher F1 score means better accuracy for the classifier.

In binary and multi-class classification, there is another method is proposed which is called as Matthews correlation coefficient (MCC). The case of unbalanced classes requires special measurement techniques such as MCC which handles unbalanced classes by using all results of predictions. The value of MCC is between -1 and +1. Positive higher values indicate better prediction results. If the MCC value is close to 0, the prediction results are random. Negative higher values indicate total disagreement between expectation and prediction results. In the case of very imbalanced classes, the F1 score can be misleading since the F1 score doesn't take into account all results of predictions in the computation of accurate measurement. Equation (2.6.9) shows how to compute MCC value.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6.5}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.6.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.6.7}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precesion + Recall} \tag{2.6.8}$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.6.9}$$

2.6.10 t-Stochastic Neighbor Embedding

Stochastic Neighbor Embedding (SNE) transforms the high dimensional data to lower dimensional data in order to fit the data in Euclidean space [22]. SNE is a technique which provides better and faster visualization of intense data items. The mapping of higher dimensional data to lower dimensional data should preserve distribution of the original data as much as possible. SNE computes the distances between data items and create conditional probabilities with these distances. Let $P(x_j|x_i)$ be the conditional probability that data item x_i would choose x_j as its neighbor. If the distance between data items x_i and x_j is relatively smaller, $P(x_j|x_i)$ will be high. A Gaussian probability density function is centered for each object such as x_i and x_j which is used to form a Gaussian distribution over the potential neighbors [23]. σ is the variance of the Gaussian that is centered at data item x_i. The conditional probability can be represented by the Equation (2.6.10). The new data items with lower dimensions can be represented as y_j and y_i. The conditional probability for new data items is $Qx_j|x_i$. Since the aim of SNE is converting higher dimensional vectors to lower dimensional vectors, the mismatched between the conditional probabilities $P(x_j|x_i)$ and $Q(x_j|x_i)$ should be minimized. The mismatched can be calculated with using relative entropy which is represented with Equation (2.6.12) where n represents number of data items.

$$P(x_j|x_i) = \frac{exp(-|x_i - x_j|^2/2\sigma^2)}{\sum exp(-|x_i - x_j|^2/2\sigma^2)} \tag{2.6.10}$$

$$Q(x_j|x_i) = \frac{exp(-|y_i - y_j|^2)}{\sum exp(-|y_i - y_j|^2)} \tag{2.6.11}$$

$$Mismatch = \sum_{i}^{n} \sum_{j}^{n} P(x_j|x_i) \log \frac{P(x_j|x_i)}{Q(x_j|x_i)} \tag{2.6.12}$$

SNE has a hyper-parameter, perplexity, which strongly affects the result of SNE [23]. In 2008, van der Maaten and Hinton [22] proposed a new approach for SNE which is named as t-SNE. The proposed method, t-SNE, aims to optimize the cost function by using different distribution (Student-t distribution) rather than Gaussian. The similarity between low dimensional data points is computed with Student-t distribution. Also, t-SNE uses symmetric SNE to reduce the cost function. The symmetric SNE

is a way to minimize the difference between the probabilities $P(x_j|x_i)$ and $Q(x_j|x_i)$. The symmetric SNE offers that instead of using the conditional probabilities, joint probabilities can be used, but joint probabilities cause a problem for the outlier data points. Since an outlier is far from other data points, joint probability has little effect on the cost function(2.6.12) which makes it difficult to determine the position of an outlier. In order to avoid the problem, Equation (2.6.13) is proposed to increase the contribution of the outliers.

$$P(x_i, x_j) = \frac{P(x_j|x_i) + P(x_i|x_j)}{2n} \tag{2.6.13}$$

2.7 Literature Survey on Word Embeddings

Continuous bag-of-words (CBOW) and skip-gram models are explained in detail in Section 2.6.1.

Table 2.2 shows the approaches and their related information which are mentioned in this study. Hyper-parameter sets of Word2vec models are indicated for each approach. Most of the studies use the skip-gram approach for Word2vec model and these studies use vector size hyper-parameter, N, as 100.

Table 2.2: Comparison of the approaches for word embedding.

Approach	**Dataset**	W	N	**Model**	**Problem**
ProtVec [6], 2015	546790 sequences of Swiss-Prot database	25	100	Skip-gram	Protein classification
Seq2vec [5], 2016	324018 sequences of Swiss-Prot database	5	250	CBOW	Protein classification
Dna2vec [24], 2017	hg38 database	10	100	Skip-gram	Relationship between Nucleotides concatenation and summing of dna2vec vectors
SMILESVec [25], 2018	A-50 dataset		100	Skip-gram	Protein classification

There are a few number of studies for word embeddings and their applications with biological sequences. Biological sequences can be treated as sentences that is proposed by Asgari and Mofrad [6] where the continuous skip-gram model is used for protein sequence embedding. Protein sequence embedding is the application of word embedding for the protein sequences where the subsequences are treated as words. The model is named as the continuous distributed representation of sequences. Also, gene sequences can be represented with the model. Asgari and Mofrad named the models as ProtVec which is for protein sequences and GeneVec which is for gene sequences. The vector representation of biological sequences which are created by word embedding technique can be used for classification of proteins, prediction of biological structures and prediction of interactions between proteins. Protein sequences are obtained from Swiss-Prot. Then, the classification of protein families is explained. The sequences of the proteins are the essential inputs for the classification process.

ProtVec is implemented to use the real-valued vectors of protein sequences which are obtained from word embedding model to train a classifier model. The accuracy of that prediction is nearly 93%. 100 dimensional feature vectors are obtained from the projection layer of the skip-gram model. Those feature vectors are inputs to support vector machine (SVM) classifier. In order to represent protein sequences, sequences are divided into subsequences and overlapping 3-mers (3-grams) are employed. Overlapping 3-gram subsequences results in three different sequences because the authors choose to generate new sentences to train the model. Asgari and Mofrad calculate sequence embeddings from the sum of subsequence embeddings. Also, t-distributed stochastic neighbor embedding is applied to the feature vectors to show biophysical and biochemical properties of proteins are distributed correctly in 2D space.

In 2016, Kimothi, Soni, Biyani, and Hogan suggested a way of constructing Word2vec models for proteomics and genomics [5]. Embedding a sequence in a lower-dimensional vector is the goal of this paper. The expectation of embedding is retrieving meaningful information from sequences and use the information for inputs to machine learning algorithms. The authors call the method as seq2vec which embeds subsequences of biological sequences into feature vectors. The seq2vec method is based on the work by Asgari [6]. The seq2vec uses 3 as the length of subsequences which can be referred to as 3-mers. Overlapping and non-overlapping subsequences are compared. The authors used another approach of Word2vec as a model to create feature vectors from protein sequences which is called as doc2vec and the doc2vec method is slightly different from Word2vec but the fundamentals of models are nearly the same. The main goal of using doc2vec is keeping the information about the orders of subsequences. In order to compute the feature vector for a sequence, the sum of feature vectors for subsequences is used. However, with this approach orders of subsequences are lost. The addressed issue can be resolved with the doc2vec model since doc2vec uses sequence and subsequences to construct the model. The distributed-memory (DM) model is the combination of CBOW model and unique document id (d_1). DBOW model is a combination of skip-gram model and the document id (d_1). After the construction of feature vectors, the authors use *k*NN as a classifier and test their doc2vec models with the results of the classifier which is used to predicts protein families. The results of seq2vec are compared with the results of the Basic Local Alignment Search Tool

(BLAST) and ProtVec. The authors claimed that the seq2vec approach is better than ProtVec.

In 2017, Ng used word embedding model Word2vec on DNA sequences [24] and the author called this approach as dna2vec. A DNA sequence is divided into subsequences and k-mer representation is used. Also, Ng mentions finding the correlation between Needleman-Wunsch [10] similarity score and the similarity of Word2vec vectors. The objective of this paper is finding the relationship between nucleotides concatenation and summation of Word2vec vectors of these nucleotides. The selected hyper-parameters of Word2vec are different from those of the study of Asgari and Mofrad [6] and Kimothi, Soni, Biyani, and Hogan [5], since the number of amino acids in a DNA sequence is smaller than the number of amino acids in protein sequences. The length of subsequences is in the range of 3 to 8. The feature vector size (N) for Word2vec is selected as 100. In addition to the training phase, Ng mentions about Needleman-Wunsch algorithm [10] which computes the similarity of biological sequences with dynamic programming. Needleman-Wunsch algorithm is a method for sequence alignment which scores global alignments from given sequences. The paper indicates that the cosine distance of k-mers is related to Needleman-Wunsch distance of the corresponding k-mers. The first step of training is splitting DNA sequences with gap characters. Gap characters are unknown amino acids in sequences and represented as "X" or "-" or "U" in the biological sequence. Then, the DNA sequence is divided with a fixed length of l which is used to represent the k value of k-mer. The divided subsequences are overlapping subsequences. The next step is training the Word2vec model with subsequences. Ng chooses the skip-gram method for Word2vec training because the skip-gram works better with infrequent subsequences. In the paper [24], context window size (W) is selected as 10 which means 10 history and 10 future subsequences are included training process for current subsequence.

Ng [24] represents two Word2vec vectors as v, w and uses Equation 2.7.1 to calculate the similarity between two these vectors. In this paper, Ng shows results that support that concatenation of the subsequences is related to summing word embeddings of these subsequences. Ng gives results about the relation of cosine similarity of Word2vec vector and Needleman-Wunsch similarity and plots the result in or-

der to show the relation. 1000 closest 8-mers are sampled to compare the results of Word2vec with Needleman-Wunsch similarity of closest 1000 8-mer subsequences.

$$sim(v, w) = \frac{v.w}{\|v\| \, \|w\|} \tag{2.7.1}$$

Öztürk, Ozkirimli, and Özgür proposed a method to find similarity of proteins by using protein's interacting ligands [25]. Ligand is an ion or molecule which binds to the central metal atom. In this paper, protein functions are predicted from ligands which have a chemical characteristic. Ligand's chemical characteristic is a known identifier of the function of the protein. Word2vec is used as a word embedding model in order to represent protein sequences as real-valued continuous vectors. Simplified Molecular Input Line Entry System (SMILES) is used to represent ligands. SMILES is a character-based representation of molecules such as ligands. Vector representations of ligands are constructed from a manually annotated and reviewed large SMILES corpus via Word2vec model. Each protein is described by using its interacting ligand vector representations. Öztürk, Ozkirimli, and Özgür named their approach as SMILESVec which use word embeddings with SMILES strings. The dataset of SMILESVec is retrieved from UniProt and ChEMBL. UniProt stores protein identifiers and sequences, but the interacting ligands are stored in ChEMBL database. The ligand SMILES are divided into 8 characters of overlapping substrings. The choice for the number of characters is based on the experiments and experiments include the range of 4-12 characters. The protein sequences are divided into 3 non-overlapping amino acids. Protein vectors are constructed from the average of vector values of subsequences. Also, other methods are assessed such as computing maximum values of each feature of subsequence vectors which is described in Equation 3.2.2 and minimum values of each feature of subsequence vectors which is described in Equation 3.2.4. Another important parameter is vector size (N) of Word2vec which is selected as 100. The computation method for Word2vec is the skip-gram. In order to compare the SMILESVec results, BLAST and ProtVec based methods are used. BLAST has e-values which can be computed from sequences and these e-values are important results since the accuracy rate of BLAST is base ground for sequence analysis. In order to measure the accuracy of the results, the clustering algorithms are used. First clustering algorithm is Transitive Clustering (TransClust) [26]. Second clustering al-

gorithm is Markov Clustering Algorithm (MCL) [27]. The sequence embeddings are fitted into TransClust [26] and Markov Clustering Algorithm (MCL). The F-measure, precision and recall scores of the results of the clustering algorithms are fetched and compared. The results indicate that TransClust has better F-measure values.

CHAPTER 3

DATASETS AND METHODS

3.1 Datasets in General

Training data for this study consist of two main parts which are positive datasets and negative datasets. Positive and negative datasets are split in order to achieve 5-fold cross validation. Datasets contain 80% training data and 20% validation data. The positive training dataset is used to generate Word2vec models. The validation dataset is used to measure the performance of Word2vec and classifier models. The UniProt Knowledgebase (UniProtKB) is a public database for the collection of annotated proteins. Each entry in the database contains the amino acid sequence and protein name. Protein sequences and names are extracted from UniProtKB/Swiss-Prot Release 2017_3. The EC Number annotations are defined in the ENZYME database (http://enzyme.expasy.org/). These manually annotated and reviewed protein sequences and EC Number annotations are obtained from UniProtKB/Swiss-Prot and ENZYME database by Dalkıran [28]. All protein sequences and names that are associated with enzyme functionalities are retrieved. Since one protein can be associated with multiple EC Number, some of the proteins are removed from datasets to avoid conflicts by Dalkıran [28].

UniProtKB/Swiss-Prot provides FASTA files. FASTA is a text-based format that bioinformatics applications use. FASTA format contains amino acid sequences, protein names, and comments about proteins. There is a unique identifier for each sequence. These identifiers include the database name.

There are 2 datasets which are taken from the work of Dalkıran [28]. The first one contains enzymes and non-enzymes. Table 3.1 shows the detail of the Level 0 dataset.

The second dataset contains the first level of Enzyme Commission classes. Table 3.2 shows the number of sequences for each class in the Level 1 dataset. Level 0 dataset is filtered by using UniRef50 [29] module which clusters proteins based on their sequence similarities.

Table 3.1: Number of proteins in Level 0 datasets.

Dataset	Number of Positive Proteins	Number of Negative Proteins
Protein Sequences	5183	13972

Table 3.2: Number of proteins in Level 1 datasets.

Dataset	Number of Positive Proteins	Number of Negative Proteins
Oxidoreductases	36822	39920
Transferases	37812	22372
Hydrolases	13440	28922
Lyases	36819	39920
Isomerases	37815	22369
Ligases	13439	28920

3.2 Methods

Biological sequences are processed and classified with libraries that are available for public. In the implementation phase, Python is used for this study. Python is a programming language which has libraries for data science such as Tensorflow [30], scikit-learn [31] and Gensim [32]. Also, Python has nice graph visualization libraries such as ggplot [33] and Matplotlib [34]. Python has different distribution versions and these versions might change the results, therefore Python version is fixed during the experiments as Python 3.7. In this study, Gensim is used for the implementation of feeding Word2vec network and fetching data from the Word2vec model. For classifications, scikit-learn library and its components (KNeigborsClassifier, GaussianNB, RandomForestClassifier, AdaboostClassifier, MLPClassifier, and SVC) are

used to classify feature vectors which are obtained from the Word2vec model. Another component of scikit-learn is TSNE which is the implementation of t-SNE algorithm. TSNE is used to visualize feature vectors in a two-dimensional space. Matplotlib library is used to convert TSNE results into a graph. In order to optimize the epoch parameter of Word2vec, training loss values are calculated. The calculated values are plotted as loss-epoch graphs with ggplot library. During the experiments, a computer with 2.40 GHz Intel 5500U processor and 8GB memory is used.

3.2.1 BLAST

BLAST has a command-line application that is available for computations. In order to calculate similarities of protein sequences, two commands of BLAST can be used. The first one is makeblastdb [35] which creates a database from fasta files. The database contains a unique identifier for each biological sequence. The second command is blastp which computes similarities between biological sequences from a given BLAST database.

3.2.2 Word2vec Model

Algorithm 3 shows the algorithm to calculate feature vectors of sequences from the real-valued vectors of subsequences by using the trained Word2vec model. The first step of the algorithm is retrieving the values of hidden nodes of the Word2vec model that correspond to subsequences. In order to form a single feature vector for protein sequence, feature vectors of subsequences are merged. There are several options to form meaningful feature vector for protein sequence. The total number of subsequences of protein sequences can be represented by n and the total number of dimensions of the feature vector can be represented by N.

Algorithm 3 Feature Vector Calculation from Word2vec Subsequences

Require: S: is a protein sequence in training dataset, s_1, s_2, s_3, ... s_n: subsequences of S, $\mathbf{v}$: is the real-valued vector of the Word2Vec model, N: is the number of dimensions of feature vector, $\mathbf{v}_{i1}$, $\mathbf{v}_{i2}$, ..., $\mathbf{v}_{iN}$: values in the dimensions of $i-th$ feature vector of the Word2vec model, **Method**: MAX, MIN, AVERAGE

```
for i ← 1 to n do
    v_i ←fetch Word2vec model's hidden node values for s_i
    sum ← sum + v_i
end for
if Method is MAX then
    for i ← 1 to n do
        maxvalues_i ← MAX(v_i1, v_i2, ..., v_iN)
    end for
    return maxvalues
else if Method is MIN then
    for i ← 1 to n do
        minvalues_i ← MIN(v_i1, v_i2, ..., v_iN)
    end for
    return minvalues
else if Method is AVERAGE then
    average ← sum/n
    return average
end if
```

One of the options is finding the maximum value of each feature of subsequences. The maximum value of a dimension of real-valued feature vectors is considered to be the best value for a feature. Therefore the maximum values are collected for each feature (v) from Word2vec vectors as shown in Equation 3.2.1. The maximum values of the first dimension are referred to as ψ_0. The maximum value of all dimensions ($\psi_0..\psi_N$) creates a vector which is referred to as ω. n is the total number of subsequences in a sequence. Equation 3.2.2 describes the method for obtaining the maximum valued

vector of the feature vector of a biological sequence.

$$\psi_0 = max([v_{00}, ..v_{n0}]) \tag{3.2.1}$$

$$\omega = \psi_0..\psi_N \tag{3.2.2}$$

The second option is retrieving the minimum values of each feature of Word2vec vectors of biological sequences. On the contrary, the minimum value of a dimension of real-valued feature vectors is considered to be the best value for a feature. Therefore the minimum values are collected for each feature (v) from Word2vec vectors as shown in Equation 3.2.3. The naming conversions are changed since calculation results are referred to in the next sections. The minimum values of the first dimension are referred to as ϕ_0. The minimum value of all dimensions ($\phi_0..\phi_N$) creates a vector which is referred to as ρ. Equation 3.2.4 describes the method for obtaining the minimum vector of the feature vector of a biological sequence.

$$\phi_0 = min([v_{00}, ..v_{n0}]) \tag{3.2.3}$$

$$\rho = \phi_0..\phi_N \tag{3.2.4}$$

The third method is averaging the feature vectors of subsequences. Since all subsequences contribute to the sequence's feature vector, a sum of vectors is a reasonable choice. The sum of the feature vectors of a biological sequence is normalized with this method. Equation 3.2.5 describes the calculation method for the normalized sum of feature vectors of a subsequence. The normalized feature vector is referred to as λ.

$$\lambda = \frac{\sum_{i=0}^{n} v_i}{n} \tag{3.2.5}$$

Algorithm 6 shows how to train and test the Word2vec model by using Algorithm 4 and Algorithm 5. In this study, Word2vec model is trained with the sequences in the positive training dataset and tested with the sequences in the positive and negative test datasets. Vector representations of the sequences in the positive and negative training

data items are used for training classifiers. Performance metrics of these classifiers are measured with vector representation of the sequences in the positive and negative test data items.

Algorithm 4 Training Word2vec Model with Subsequences

Require: β_p: Number of positive sequences, β_1: Number of positive training data, β_2: Number of positive test data, S_1, S_2 **...,** S_{β_1}: are the protein sequences in training dataset, s_1, s_2, s_3, **...** s_n: subsequences of S, $\mathbf{v}$: is the real-valued vector of Word2Vec, ζ: Word2vec vocabulary size

for $i \leftarrow 1$ to β_1 **do**

Add s_1, s_2, s_3, ... s_n of S_i to Word2vec vocabulary

end for

Train Word2vec with using the parameters N and W

for $i \leftarrow 1$ to β_1 **do**

for all s_1, s_2, s_3, ... s_n of S_i **do**

$\mathbf{v} \leftarrow$ fetch Word2vec hidden node values for s

end for

Calculate feature vectors of S_i with using Algorithm3 where subsequences of S_i are s_1, s_2, s_3, ... s_n

end for

3.2.3 Comparison of BLAST-Word2vec

Word2vec is used to form the feature vector of biological sequence. The distance between feature vectors can be calculated and can be interpreted as the similarity between the proteins. In order to convert the distance values to similarities, Equation 2.7.1 is used. The cosine similarity is a method that is used to calculate similarities in several studies [24] [25]. Similarities of sequences depend on the hyper-parameters of Word2vec. In order to illustrate the distance, some of the proteins are chosen randomly. Table 3.3 shows identifiers of these proteins and distances between these proteins. Cosine similarities and BLAST scores show that there is a correlation between BLAST and vector representations of Word2vec. BLAST result doesn't contain the similarity between some proteins such as A5FZ54-Q49VI3, since BLAST

Algorithm 5 Testing the Results of Classifiers

Require: β_2: Number of positive test data, β_n: Number of negative data, β_3: Number of negative training data, β_4: Number of negative test data, S_1**,** S_2 **...,** $S_{\beta_2+\beta_n}$: are the protein sequences in dataset, s_1**,** s_2**,** s_3**, ...**s_n: subsequences of S, $\mathbf{v}$: is the real-valued vector of Word2Vec, ζ: Word2vec vocabulary size

for $i \leftarrow 1$ to $\beta_2 + \beta_n$ **do**

for all s_1, s_2, s_3, ... s_n of S_i **do**

$\mathbf{v} \leftarrow$ fetch Word2vec hidden node values for s

end for

Calculate feature vectors of S_i with using Algorithm3 where subsequences of S_i are s_1, s_2, s_3, ... s_n

end for

Algorithm 6 Training and Test of the Word2vec Model

Require: $positive\_sequences$: are the sequences of positive data, $negative\_sequences$: are the sequences of negative data,$positive\_training\_sequences$: are the sequences of positive training dataset, $positive\_test\_sequences$: are the sequences of positive test dataset, $negative\_training\_sequences$: are the sequences of negative training dataset, $negative\_test\_sequences$: are the sequences of negative test dataset, β: Number of sequences, β_p: Number of positive sequences, β_n: Number of negative sequences β_1: Number of positive training data, β_2: Number of positive test data, β_3: Number of negative training data, β_4: Number of negative test data

Use Algorithm 4 with $positive\_training\_sequences$, β_1, β_2 and β_p to get the list of $positive\_training\_vectors$

Use Algorithm 5 with $positive\_test\_sequences$, $\beta_2, \beta_3, \beta_4$, and β_n to get the list of $positive\_test\_vectors$, $negative\_train\_vectors$ and $negative\_test\_vectors$

Fit the labeled data lists ($positive\_training\_vectors$ and $negative\_training\_vectors$) into classifiers

Measure performance metrics of classifiers with unlabeled data lists ($positive\_test\_vectors$ and $negative\_test\_vectors$)

has a threshold value (E-value) to ignore irrelevant results. The hyper-parameters for Table 3.3 are 5 for window size (W), 50 for feature vector size (N), non-overlapping, 3 for the length of subsequences value (l) and calculation method for feature vector of sequence is the average value of feature vectors which is described in Equation3.2.5.

Table 3.3: Relation of BLAST score and cosine similarity of vector representations of Word2vec model.

Protein Id	**Protein Id**	**Cosine Similarity**	**BLAST Score**
A5FZ54	Q6MGM7	0.9825	68.6970
A5FZ54	Q2VZN2	0.9937	79.6140
A5FZ54	Q49VI3	0.7844	Too Low

In order to compare the results of Word2vec and BLAST, 1000 most similar proteins with each other and 1000 most dissimilar proteins are chosen. BLAST finds similarities between proteins but the E-Value leaves some protein-protein similarities out of the results. Since BLAST doesn't compute all protein pairs, Word2vec results are sorted from most similar to most dissimilar. The sorted results of Word2vec is searched inside BLAST results. If there isn't any match, the Word2vec result is ignored. The relation between results are needed to be computed. Pearson correlation coefficient (PCC) is used to compute correlation. Pearson correlation coefficient is a computation method for linear correlation between two variables which can be referred to as X and Y. The correlation value between +1 and -1 which are the highest correlation values. Higher correlation value indicates positive linear correlation. Equation 3.2.6 shows the calculation method for Pearson correlation coefficient where cov is the covariance and σ_x is the standard deviation of X and σ_y is the standard deviation of Y. Covariance is the calculation method for joint variability of two random variables. Covariance is positive, if the higher values of BLAST-Word2vec similarity lists correspond. Equation 3.2.7 shows the formula of covariance calculation where $E[X]$ is the expected mean of the given list X. The expected mean indicates that a sample set of the list is taken to calculate average value. Algorithm 7 shows the pseudocode for comparison of the BLAST scores and similarities of vector representations of Word2vec model.

Algorithm 7 Comparison of BLAST-Word2vec similarities

Require: p: is the protein id, ζ: is the size of the Word2vec vocabulary, (p_1, p_2 $\ldots p_\zeta$): are the protein ids which are fetched from Word2vec vocabulary, $\mathbf{v}$: is the real-valued vector of Word2Vec, b: is the BLAST similarity, z: is the number of results of the BLAST scores, (b_1, b_2, $\ldots b_z$): are the BLAST scores, a: is the number of comparison between Word2vec-BLAST, q: is the sorted similarity list which contains two protein ids (p_i and p_j), o: is the most similar protein's score list, r: is the most dissimilar protein's score list

for $i \leftarrow 1$ to ζ **do**
  for $j \leftarrow 1$ to ζ **do**
    if $i \neq j$ **then**
      Calculate the similarity between the $\mathbf{v}$ value of p_i and $\mathbf{v}$ value of p_j using the cosine similarity which is given in Equation (2.7.1)
    end if
  end for
end for
$q \leftarrow$ Sort the cosine similarities of proteins
for all q **do**
  if number of iterations $\leq a$ **then**
    $o_i \leftarrow$ cosine similarity of the Word2vec vector
    $o_i \leftarrow$ the BLAST similarity score in (b_1, b_2, $\ldots b_z$)
  end if
end for
$q \leftarrow$ Reverse the cosine similarities of proteins
for all q **do**
  if number of iterations $\leq a$ **then**
    $r_i \leftarrow$ cosine similarity of the Word2vec vector
    $r_i \leftarrow$ the BLAST similarity score in (b_1, b_2, $\ldots b_z$)
  end if
end for
Calculate the deviation for BLAST scores and cosine similarities of o
Calculate the deviation for BLAST scores and cosine similarities of r
Calculate the Pearson correlation between BLAST scores and cosine similarities by using the Equation 3.2.6

Table 3.4 shows 9 columns of which first 4 columns are hyper-parameters for Word2vec. First column is window size (W) which has three different values as 5, 10 and 25. Second column is vector size (N) for Word2vec which has three different values as 100, 150 and 200. Third column is l value for k-mer. Fourth column is the calculation method for feature vector of biological sequence. Fifth column indicates average similarity value of Word2vec and Blast similarities of most similar proteins. Seventh column indicates average similarity value of most dissimilar proteins. Sixth and eighth columns are deviation values. Last column is Pearson correlation coefficient value of most similar and most dissimilar protein similarities.

$$p_{X,Y} = \frac{cov(X,Y)}{\sigma_x \sigma_y} \tag{3.2.6}$$

$$cov(X,Y) = E[(X - E[X])(Y - E[Y])] \tag{3.2.7}$$

Table 3.4: Comparison of Word2vec-BLAST similarity scores for Level 0 dataset.

W	N	l	**Method**	**Avg Sim**	**Sim Dev**	**Avg Dissim**	**Dissim Dev**	**Corr**
5	50	3	sum	95.90	3.18	69.87	6.22	1.00
10	50	3	sum	95.75	3.37	69.30	5.96	1.00
25	50	3	sum	95.82	3.23	68.84	5.85	1.00
5	100	3	sum	96.02	3.02	69.78	6.26	1.00
10	100	3	sum	95.70	3.36	47.98	4.31	0.99
25	100	3	sum	95.80	3.39	50.07	4.16	0.99
5	150	3	sum	96.06	2.98	69.37	6.12	1.00
10	150	3	sum	96.00	3.05	69.56	6.45	1.00
25	150	3	sum	96.00	3.04	68.58	5.90	1.00
5	200	3	sum	96.06	2.99	68.97	6.02	1.00
10	200	3	sum	96.05	2.98	69.33	6.31	1.00
25	200	3	sum	96.04	2.98	68.17	5.65	1.00
5	50	5	sum	95.70	3.33	48.01	4.66	0.99
10	50	5	sum	95.73	3.43	47.87	4.23	0.99

25	50	5	sum	95.81	3.37	49.99	4.19	0.99
5	100	5	sum	95.70	3.32	47.86	4.16	0.99
10	100	5	sum	**96.74**	**2.25**	**61.89**	**2.04**	**1.00**
25	100	5	sum	**96.77**	**2.27**	**61.94**	**1.74**	**1.00**
5	150	5	sum	95.69	3.33	47.81	4.21	0.99
10	150	5	sum	95.71	3.46	47.95	4.15	0.99
25	150	5	sum	95.76	3.49	49.60	3.89	0.99
5	200	5	sum	95.67	3.37	48.42	4.61	0.99
10	200	5	sum	95.74	3.43	48.12	4.18	0.99
25	200	5	sum	95.80	3.42	49.49	3.71	0.99
5	50	5	max	94.90	4.76	56.83	4.21	1.00
10	50	5	max	95.17	3.97	54.39	4.07	1.00
25	50	5	max	94.79	4.57	52.67	3.56	1.00
5	100	5	max	95.28	3.78	55.63	4.51	1.00
10	100	5	max	94.58	5.03	55.41	3.50	1.00
25	100	5	max	94.02	5.86	53.16	3.80	0.99
5	150	5	max	95.16	3.83	57.30	4.30	1.00
10	150	5	max	94.98	4.58	55.39	4.52	0.99
25	150	5	max	94.35	5.26	53.25	3.45	0.99
5	200	5	max	95.23	3.73	56.21	4.21	1.00
10	200	5	max	95.10	4.16	55.41	4.16	1.00
25	200	5	max	94.44	5.24	53.89	3.45	0.99
5	50	5	min	95.08	4.29	54.91	4.22	1.00
10	50	5	min	94.21	5.77	55.28	4.15	0.99
25	50	5	min	93.76	5.62	52.18	4.37	0.99
5	100	5	min	95.21	4.03	56.75	4.50	1.00
10	100	5	min	94.97	4.55	55.27	4.26	1.00
25	100	5	min	94.30	4.93	53.13	4.18	0.99
5	150	5	min	95.21	3.97	54.96	4.27	1.00
10	150	5	min	95.04	4.45	54.66	4.21	1.00
25	150	5	min	94.23	5.63	53.18	3.57	0.99

5	200	5	min	95.29	3.74	56.30	4.33	1.00
10	200	5	min	94.91	4.62	54.88	3.84	1.00

3.2.4 Loss Value

The loss value of the training Word2vec model is an important indicator which shows the network of convergence of Word2vec model. A good epoch value for Word2vec can be determined based on the loss value which is computed during the training process by using Equation 3.2.8 where total number of subsequences is n and window size is W. Weights between input layer and hidden layer are multiplied with the weights between hidden layer and output layer to calculate the total weight values of nodes in the output layer and u_j is referred as the total weight value of j-th node of the output layer.

$$Loss = -\sum_{i=0}^{W} u_i + W \cdot \log \sum_{j=1}^{n} \exp(u_j) \tag{3.2.8}$$

3.2.5 Support Vector Machine

Correct values for the γ parameter and C value depends on the experiments. These hyper-parameters can be adjusted using the grid search method. The grid search could be implemented as in Algorithm 8. From the grid search algorithm 8, MCC scores are calculated and compared to find the best hyper-parameters.

Algorithm 8 Grid Search for Hyper-parameters of SVM

Require: γ: is the parameter of the Gaussian radial basis function of SVM, C: is the parameter for the soft margin cost function of SVM, $\mathbf{v}$: is the real-valued vector of Word2Vec model, N: is the number of dimensions of feature vector, n: number of training data

for C in range $(1e-2, 1, 1e2)$ **do**

for γ in range $(1e-1, 1, 1e1)$ **do**

Fit SVM with training data $\{\mathbf{v}_0, \mathbf{v}_1, ..\mathbf{v}_n\}$

Get SVM results

Compute MCC score with using Equation2.6.9

end for

end for

CHAPTER 4

RESULTS AND DISCUSSION

Protein sequences are embedded by using Word2vec skip-gram model. The initial experiments are mainly about the selection of the Word2vec model's hyper-parameters such as epoch value, window size (W), vector size (N), and length of subsequences (l). In order to find a good epoch value, the training loss value of Word2vec model is computed during the training process of Word2vec model. Other hyper-parameters of the Word2vec are selected based on the results of Word2vec-BLAST comparison which is explained section 3.2.3.

The classifier parameters are selected according to the experiments which are made with different hyper-parameters of classifiers. For each classifier, there could be a set of hyper-parameters such as the k value of kNN or γ value of the SVM algorithm. Also, t-SNE algorithm is used to project the feature vectors into 2D space to visualize the distribution of the representation of the biological sequences.

4.1 Results

4.1.1 Loss Value

For the epoch value, there is a trade-off between the training time and the accuracy of vector representations of biological sequences. Figure 4.1, Figure 4.2, Figure 4.3, and Figure 4.4 show that the Loss-Epoch plots for different epoch values. Blue lines indicate calculated loss value for the end of an epoch. The orange and dashed lines indicate the average of the last 10 loss values in order to show the change in loss value. The last loss values for Figure4.2, Figure 4.3, and Figure 4.4 don't show a

significant decline which means Word2vec model is trained well enough to minimize the loss value. For machine learning applications, training a model has consequences such as learning the details of the dataset that is called overfitting. In order to avoid overfitting and train a converged network, binary search approach is applied to epoch values. The optimal loss value is found where the epoch value is 180 which is almost the mid-value of 100 and 250. Figure 4.2 indicates that loss values are nearly equal to the loss values of other experiments with higher epoch values.

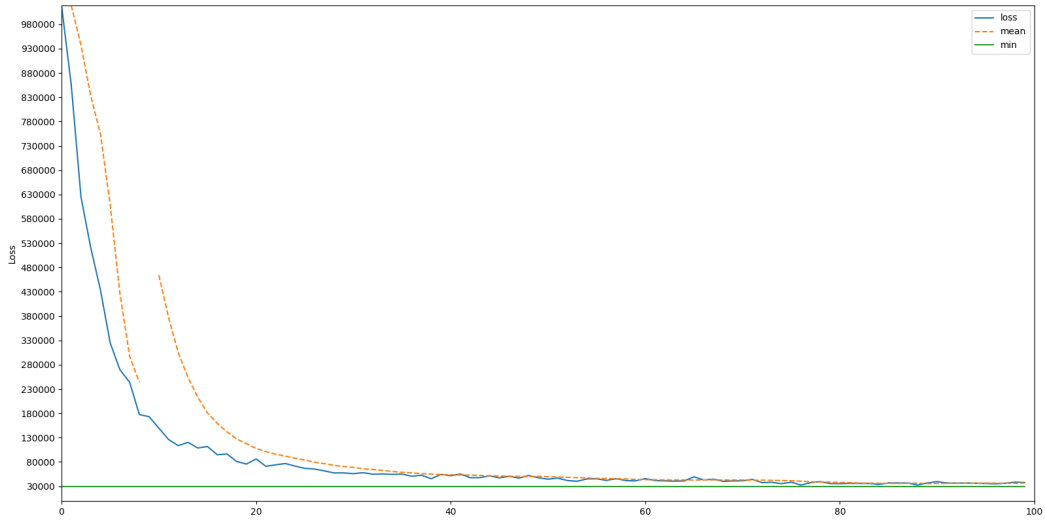

Figure 4.1: The plot of training loss values against epoch during the training of the Word2vec model where epoch value is given as 100.

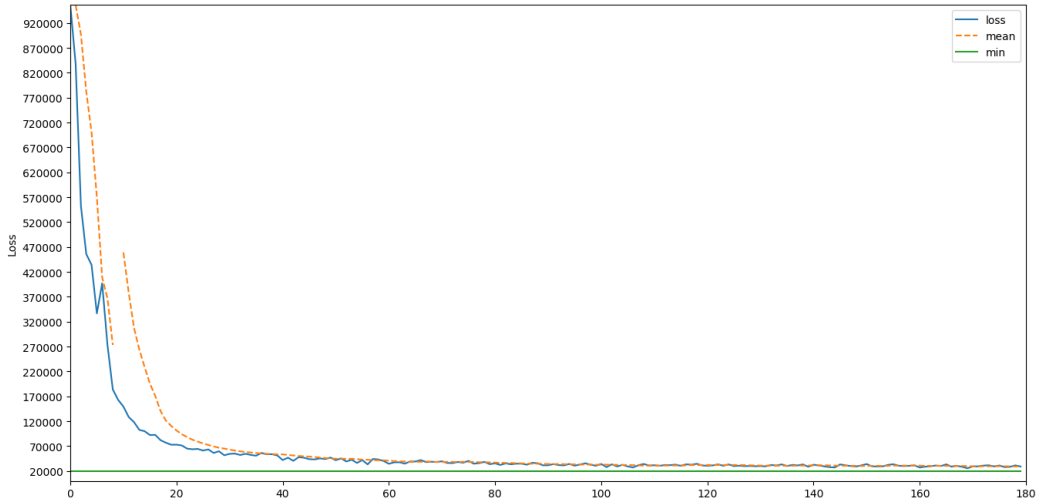

Figure 4.2: The plot of training loss values against epoch during the training of the Word2vec model where epoch value is given as 180.

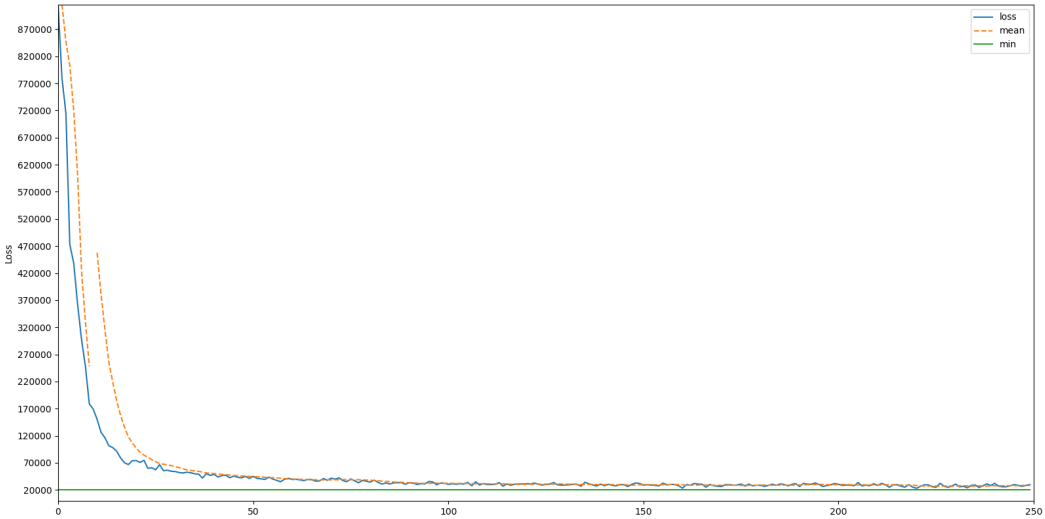

Figure 4.3: The plot of training loss values against epoch during the training of the Word2vec model where epoch value is given as 250.

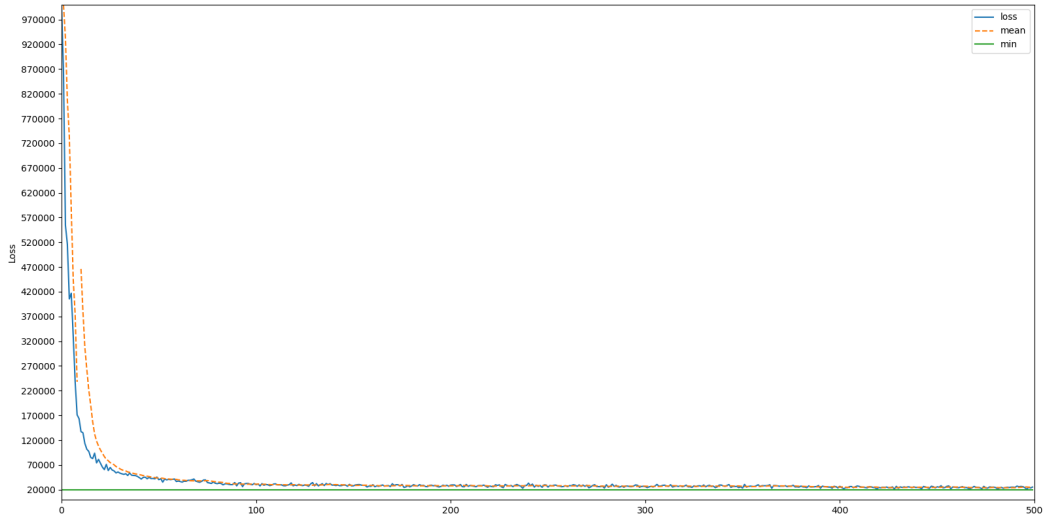

Figure 4.4: The plot of training loss values against epoch during the training of the Word2vec model where epoch value is given as 500.

4.1.2 Enzyme Commission Numbers Level 0 Results

The dataset for level 0 contains enzyme and non-enzyme protein sequences. Table 4.1 shows 10 columns which first 4 columns are hyper-parameters for Word2vec. The other columns are accuracy values for classifiers. The accuracy value is calculated from the results of 5-fold cross-validation by getting the average. The average accuracy values of the *k*NN, SVM, Naive Bayes and Random Forest classifiers are shown from fifth column to eighth column. The last 2 columns are calculated for the averaging method (sum) which is defined in Algorithm 3. The accuracy values don't always reflect the success of a classifier, therefore other performance measurement methods are given in Table 4.2 and Table 4.3. Since the averaging calculation method performs better than other methods, the rest of the results are shown for the averaging method with Word2vec hyper-parameters.

The results of the level 0 indicate hyper-parameters of Word2vec and l value of k-mer should be selected carefully. The vector size (N) affects the results directly. 100 dimensions for the N gives better results for the dataset. Also, the optimum l value for k-mer is 5. There is another parameter, window size (W) which affects the results. The best result from the classifiers indicates W value should be 10, but there is a close

result that has the W value as 25. Since the hyper-parameters are nearly fixed, the experiments on first level classes are done with these two sets of hyper-parameters.

The classifiers have hyper-parameters which are selected based on the performance of classifiers. There are 4 classifiers that are performed with different parameters. The performances of classifiers are measured based on all of the hyper-parameter sets of Word2vec. The grid search method is applied for the hyper-parameters. The first classifier is *k*NN which is executed with 3, 5, 7, 9, 13, 17, 25, and 35 as the k value. The Random Forest classifier performs with several number of estimators such as 10, 20, 40, 60, 70, 100 and 140. The same set of the estimators are used for the AdaBoost classifier to optimize the AdaBoost's estimator parameter. The last classifier is SVM which has more complex hyper-parameters such as kernel method, C and γ value.

Table 4.1: Accuracy values for each classifier with different hyper-parameters of Word2vec model.

W	N	l	**Method**	***k*NN**	**SVM**	**Naive**	**RF**	**Ada Boost**	**ANN**
5	50	3	sum	94.76	96.15	83.99	93.11	88.58	92.22
10	50	3	sum	95.43	96.50	86.84	94.07	90.40	93.48
25	50	3	sum	95.96	96.31	89.47	94.49	91.87	94.64
5	100	3	sum	94.31	96.72	81.60	92.76	88.87	93.74
10	100	3	sum	90.89	95.71	72.34	94.27	85.58	94.08
25	100	3	sum	89.79	94.43	71.20	92.33	82.55	91.81
5	150	3	sum	93.86	95.85	79.85	91.94	88.23	94.49
10	150	3	sum	94.30	96.40	83.27	93.21	89.58	94.41
25	150	3	sum	94.83	95.77	87.30	93.80	90.90	94.85
5	200	3	sum	93.40	96.31	78.32	91.77	88.36	94.55
10	200	3	sum	93.88	95.73	82.07	92.50	88.85	94.52
25	200	3	sum	94.41	95.48	86.43	93.52	90.93	94.79
5	50	5	sum	92.74	95.41	81.71	95.06	87.79	94.96
10	50	5	sum	91.48	95.16	73.05	94.32	85.78	93.41
25	50	5	sum	90.15	93.91	71.93	92.54	81.45	91.50

5	100	5	sum	92.48	96.19	82.63	95.23	87.81	95.40
10	100	5	sum	**97.88**	**98.75**	**92.57**	**98.06**	**97.08**	**98.77**
25	100	5	sum	**96.98**	**97.44**	**92.67**	**97.31**	**95.48**	**98.05**
5	150	5	sum	92.42	96.37	82.43	95.19	88.07	95.67
10	150	5	sum	90.90	95.91	72.64	94.12	86.26	94.16
25	150	5	sum	89.78	94.58	70.55	92.44	82.87	91.48
5	200	5	sum	92.49	96.50	82.66	95.31	88.65	95.68
10	200	5	sum	91.00	95.88	72.63	94.11	86.22	94.02
25	200	5	sum	89.50	94.76	70.37	92.42	82.88	91.53
5	50	5	max	96.71	96.32	95.09	96.46		
10	50	5	max	96.32	96.40	94.39	96.45		
25	50	5	max	96.04	96.21	94.71	96.10		
5	100	5	max	96.76	96.58	95.36	96.70		
10	100	5	max	96.38	96.35	95.09	96.62		
25	100	5	max	96.19	96.13	95.01	96.45		
5	150	5	max	96.72	96.64	95.15	96.80		
10	150	5	max	96.71	96.86	95.19	96.79		
25	150	5	max	96.24	96.55	94.83	96.57		
5	200	5	max	96.86	96.68	95.14	96.89		
10	200	5	max	96.63	96.87	94.82	96.78		
25	200	5	max	96.27	96.66	94.78	96.66		
5	50	5	min	96.88	96.87	95.40	96.33		
10	50	5	min	96.28	96.37	94.45	96.27		
25	50	5	min	95.75	96.08	94.91	96.45		
5	100	5	min	96.63	96.61	95.52	96.61		
10	100	5	min	96.47	96.62	95.08	96.77		
25	100	5	min	95.96	96.37	95.08	96.71		
5	150	5	min	96.88	96.83	95.38	96.76		
10	150	5	min	96.46	96.68	94.70	96.59		
25	150	5	min	96.13	96.42	95.29	96.67		
5	200	5	min	96.81	96.80	95.24	96.78		

10	200	5	min	96.60	96.77	95.09	96.88		

Table 4.2: F1 Scores for each classifier with different hyper-parameters of Word2vec model.

W	N	l	**Method**	***k*NN**	**SVM**	**Naive**	**RF**	**Ada Boost**	**ANN**
5	50	3	sum	0.95	0.96	0.84	0.93	0.89	0.92
10	50	3	sum	0.95	0.97	0.87	0.94	0.90	0.94
25	50	3	sum	0.96	0.96	0.89	0.95	0.92	0.94
5	100	3	sum	0.94	0.97	0.82	0.92	0.89	0.94
10	100	3	sum	0.91	0.96	0.73	0.94	0.85	0.94
25	100	3	sum	0.90	0.94	0.72	0.92	0.82	0.92
5	150	3	sum	0.94	0.96	0.80	0.92	0.88	0.94
10	150	3	sum	0.94	0.96	0.83	0.93	0.89	0.94
25	150	3	sum	0.95	0.96	0.87	0.94	0.91	0.95
5	200	3	sum	0.93	0.97	0.79	0.91	0.88	0.94
10	200	3	sum	0.94	0.96	0.82	0.92	0.89	0.95
25	200	3	sum	0.94	0.96	0.86	0.93	0.91	0.95
5	50	5	sum	0.93	0.95	0.82	0.95	0.88	0.95
10	50	5	sum	0.91	0.95	0.73	0.94	0.86	0.93
25	50	5	sum	0.90	0.94	0.72	0.92	0.81	0.91
5	100	5	sum	0.93	0.96	0.83	0.95	0.88	0.96
10	100	5	sum	**0.98**	**0.99**	**0.93**	**0.98**	**0.97**	**0.99**
25	100	5	sum	**0.97**	**0.97**	**0.93**	**0.97**	**0.95**	**0.98**
5	150	5	sum	0.92	0.96	0.83	0.95	0.88	0.96
10	150	5	sum	0.91	0.96	0.73	0.94	0.86	0.94
25	150	5	sum	0.90	0.95	0.71	0.92	0.82	0.91
5	200	5	sum	0.93	0.96	0.83	0.95	0.88	0.96
10	200	5	sum	0.91	0.96	0.73	0.94	0.86	0.94
25	200	5	sum	0.90	0.95	0.71	0.92	0.83	0.92

Table 4.3: MCC Scores for each classifier with different hyper-parameters of Word2vec model.

W	N	l	**Method**	***k*NN**	**SVM**	**Naive**	**RF**	**Ada Boost**	**ANN**
5	50	3	sum	0.86	0.90	0.59	0.82	0.70	0.80
10	50	3	sum	0.89	0.91	0.66	0.85	0.75	0.83
25	50	3	sum	0.90	0.91	0.73	0.86	0.79	0.86
5	100	3	sum	0.86	0.92	0.55	0.81	0.71	0.84
10	100	3	sum	0.78	0.89	0.34	0.86	0.63	0.85
25	100	3	sum	0.75	0.86	0.31	0.81	0.55	0.80
5	150	3	sum	0.84	0.89	0.52	0.79	0.70	0.86
10	150	3	sum	0.85	0.91	0.59	0.82	0.73	0.86
25	150	3	sum	0.87	0.89	0.67	0.84	0.76	0.87
5	200	3	sum	0.83	0.91	0.49	0.79	0.70	0.86
10	200	3	sum	0.84	0.89	0.56	0.81	0.71	0.86
25	200	3	sum	0.86	0.88	0.66	0.83	0.76	0.87
5	50	5	sum	0.82	0.89	0.55	0.88	0.69	0.88
10	50	5	sum	0.79	0.88	0.35	0.86	0.64	0.84
25	50	5	sum	0.75	0.85	0.32	0.81	0.52	0.79
5	100	5	sum	0.82	0.91	0.58	0.88	0.70	0.89
10	100	5	sum	**0.95**	**0.97**	**0.82**	**0.95**	**0.93**	**0.97**
25	100	5	sum	**0.93**	**0.94**	**0.82**	**0.93**	**0.89**	**0.95**
5	150	5	sum	0.82	0.91	0.58	0.88	0.70	0.89
10	150	5	sum	0.78	0.90	0.36	0.85	0.65	0.86
25	150	5	sum	0.75	0.87	0.30	0.81	0.56	0.79
5	200	5	sum	0.82	0.92	0.59	0.88	0.72	0.90
10	200	5	sum	0.78	0.90	0.37	0.85	0.65	0.86
25	200	5	sum	0.74	0.87	0.30	0.81	0.56	0.80

Table 4.4 shows all results of *k*NN with the Word2vec hyper-parameters where N is 100, W is 10, l value of k-mer is 5 and vector calculation for a sequence is averaging

method. The best k value is 3 for the given Word2vec parameter set. The performance measurements are shown for different SVM hyper-parameters in Table 4.5. AdaBoost and Random Forest Classifiers have an important parameter which is the number of estimators. Table 4.6 and Table 4.7 show the F1 and MCC scores for the different estimator values.

Table 4.4: MCC and F1 Scores for different k values of kNN.

k	**MCC Score**	**F1 Score**
3	**0.95**	**0.98**
5	0.94	0.98
7	0.94	0.98
9	0.94	0.97
13	0.93	0.97
17	0.92	0.97
25	0.91	0.96
35	0.89	0.96

Table 4.5: MCC and F1 scores for different hyper-parameter values of SVM classifier.

kernel	C	γ	**MCC Score**	**F1 Score**
linear	1.00	auto	0.93	0.97
poly	1.00	auto	0.70	0.87
rbf	0.01	0.10	0.86	0.94
rbf	0.01	1.00	0.85	0.94
rbf	0.01	10.00	0.00	0.62
rbf	1.00	0.10	**0.95**	**0.98**
rbf	1.00	1.00	**0.97**	**0.99**
rbf	1.00	10.00	0.75	0.89
rbf	100.00	0.10	**0.97**	**0.99**
rbf	100.00	1.00	**0.97**	**0.99**
rbf	100.00	10.00	0.77	0.90
sigmoid	1.00	auto	0.91	0.96

Table 4.6: MCC and F1 scores for different estimator values of Random Forest classifier.

Number of Estimators	**MCC Score**	**F1 Score**
10	0.94	0.97
20	0.95	0.98
40	0.95	0.98
60	0.95	0.98
70	**0.95**	**0.98**
100	**0.95**	**0.98**
140	**0.95**	**0.98**

Table 4.7: MCC and F1 scores for different estimator values of AdaBoost classifier.

Number of Estimators	**MCC Score**	**F1 Score**
10	0.85	0.94
20	0.87	0.95
40	0.90	0.96
60	0.91	0.96
70	0.91	0.97
100	0.92	0.97
140	**0.93**	**0.97**

In order to understand the affects of classifier's hyper-parameters, Table 4.4, Table 4.6 and Table 4.7 can be inspected. Figure 4.5 shows the *k*NN classifier's performance for the k values. Figure 4.6 and Figure 4.7 show the Random Forest and AdaBoost classifiers' performance indicators for different estimator values. The results indicate that lower k value for *k*NN produces better results. On the contrary, higher estimator value for AdaBoost and Random Forest produces better results. For the SVM classifier, higher C values with RBF kernel result better and the performance indicators are shown in the Table 4.5.

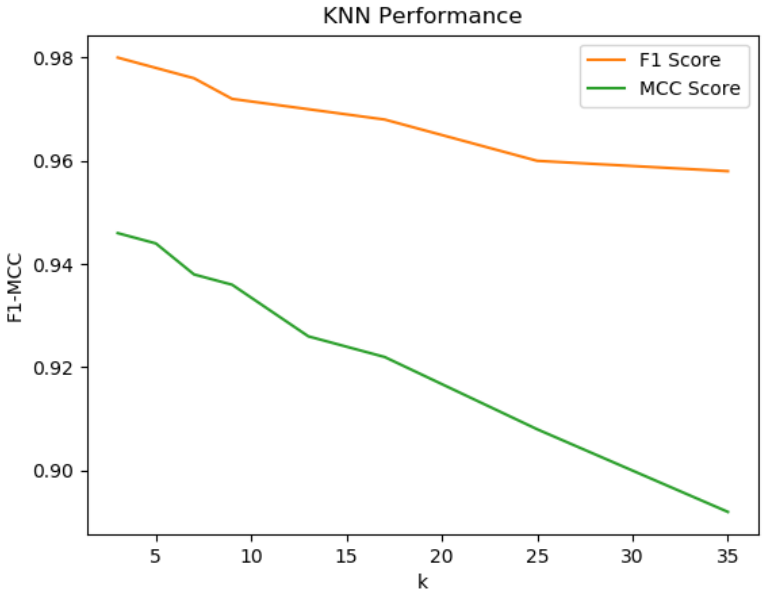

Figure 4.5: kNN performance scores with the k values.

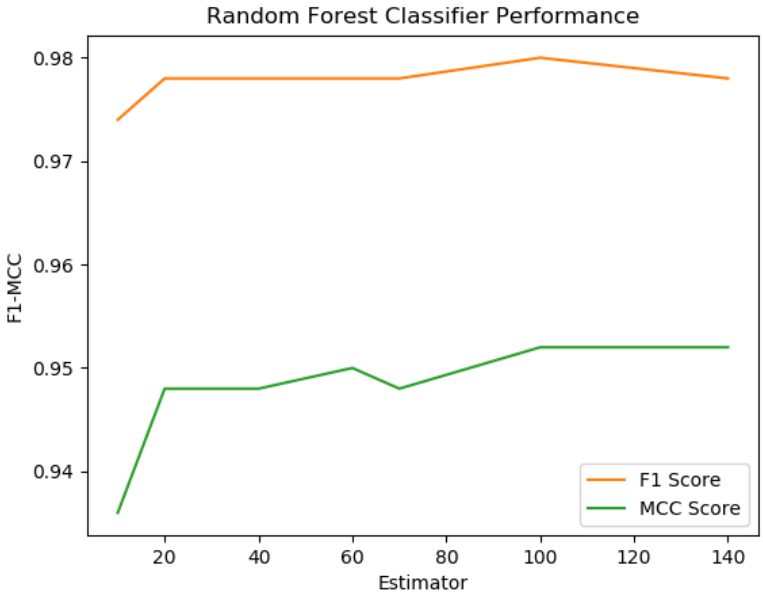

Figure 4.6: Random Forest classifier performance scores with different number of estimator values.

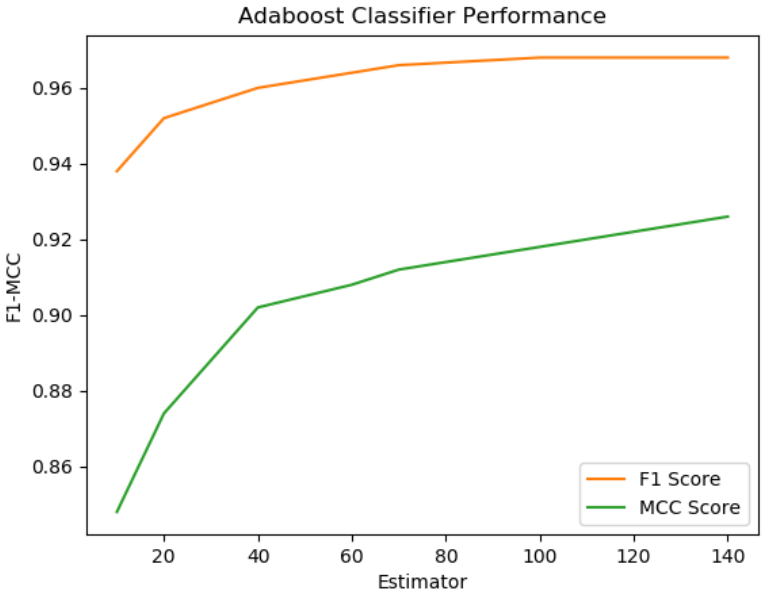

Figure 4.7: AdaBoost classifier performance with different number of estimator values.

The feature vectors are obtained from Word2vec model with the best hyper-parameter set where N is 100, W is 10, l is 5 and vector calculation for a sequence is averaging method. In order to visualize the distribution of vectors, t-SNE algorithm is used to project high dimensional vectors to two-dimensional vectors. Figure 4.8 shows the projection of data items in training dataset by using t-SNE algorithm. We applied 5-fold cross-validation, therefore there are 5 figures which represent the feature vector representation of the dataset. The points with red color correspond to non-enzyme sequences while points with blue color represent the feature vectors of enzyme sequences. Distribution of the data points indicates that data items can be classified by using vector representation of Word2vec model.

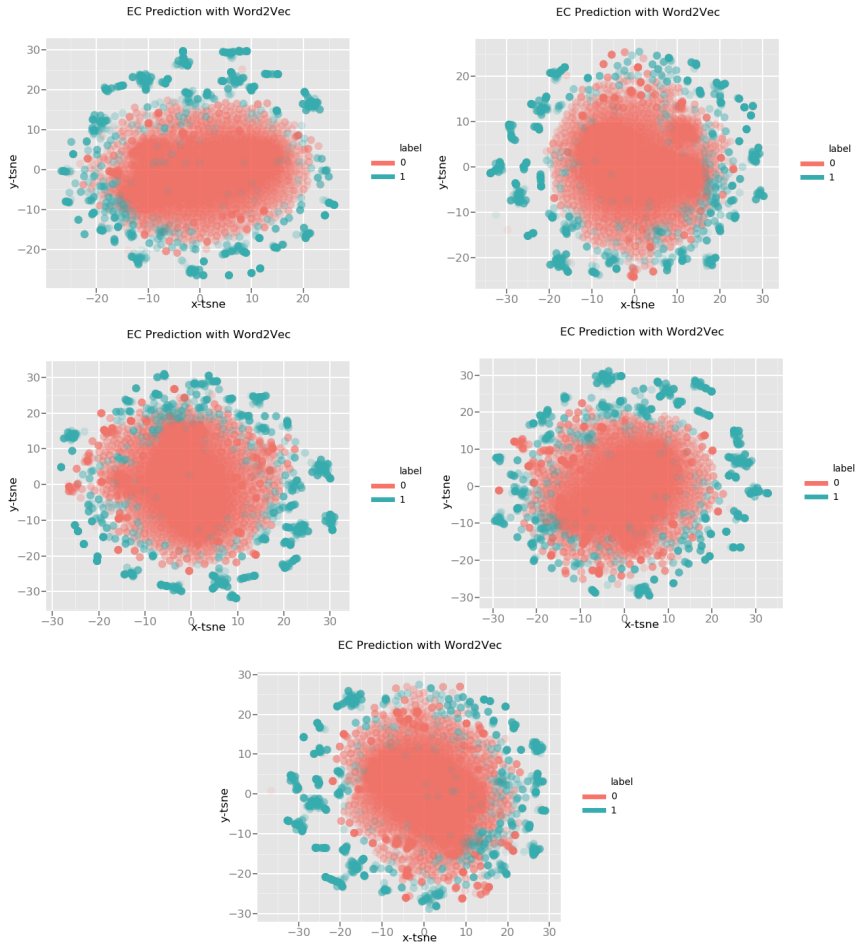

Figure 4.8: Vector representations of Word2vec model is visualized by using t-SNE algorithm to project vectors onto 2D space. Since 5-fold cross-validation is applied, there are five figures.

4.1.3 Enzyme Commission Numbers Level One Results

The studies which are conducted with the enzyme and non-enzyme dataset show that Word2vec hyper-parameters can be optimized. Although one parameter set results better than others, we decided to continue experiments with the first two best hyper-parameter sets where the first one contains 100 as N, 10 as W, 5 for l and averaging method for vector representation calculation of sequences and the second set contains the same hyper-parameters except W parameter which is 25. Table 3.2 shows the details about the number of proteins for each class, but the dataset are randomly

sampled to reduce training time of the Word2vec model. In order to sample a balanced dataset, 1000 proteins from the positive and the negative datasets are chosen. Table 4.9 contains the Word2vec-BLAST similarity comparison for these hyper-parameter sets and averaging method. The corresponding scientific names of classes are given in Table 4.8 below.

Table 4.8: Level 1 classes and scientific names of these classes.

Class	**Scientific Name**
First Class	Oxidoreductases
Second Class	Transferases
Third Class	Hydrolases
Fourth Class	Lyases
Fifth Class	Isomerases
Sixth Class	Ligases

Table 4.9: Word2vec-Blast similarity for first level of EC classes with averaging method.

	W	N	l	**Avg Sim**	**Sim Dev**	**Avg Dissim**	**Dissim Dev**	**Corr**
Oxidoreductases	10	100	5	99.97	0.07	50.26	6.68	0.99
	25	100	5	**99.97**	**0.07**	**39.81**	**1.58**	**1.00**
Transferases	10	100	5	98.77	1.47	49.39	3.42	1.00
	25	100	5	**98.81**	**1.43**	**41.59**	**3.31**	**1.00**
Hydrolases	10	100	5	99.65	0.42	49.66	4.98	1.00
	25	100	5	**99.66**	**0.39**	**41.28**	**3.81**	**1.00**
Lyases	10	100	5	100.00	0.01	47.80	6.78	0.99
	25	100	5	**100.00**	**0.01**	**40.90**	**3.46**	**1.00**
Isomerases	10	100	5	100.00	0.01	51.11	8.14	0.99
	25	100	5	**100.00**	**0.01**	**38.91**	**3.27**	**1.00**
Ligases	10	100	5	100.00	0.02	45.43	2.60	1.00
	25	100	5	**100.00**	**0.02**	**38.08**	**3.16**	**1.00**

For the best two hyper-parameter set, classifier performances are measured. Table 4.10 shows the accuracy values for classifiers. In order to show classifier performances for EC classes, first level of EC classes are shown. Table 4.11 and Table 4.12 show F1 and MCC score of first level classes. Section 4.1.2 contains the level 0 results where one of the hyper-parameter sets produces better results, but another hyper-parameter set performs as good as the best hyper-parameter set. For first level results, the another hyper-parameter set where W is 25 and N is 100 performs better for all of the classes.

Table 4.10: Accuracy values for each classifier with different hyper-parameter sets of Word2vec model.

	W	N	l	***k*NN**	**SVM**	**Naive**	**RF**	**Ada Boost**	**ANN**
Oxidoreductases	10	100	5	82.85	83.14	84.31	86.25	83.93	84.24
	25	100	5	82.64	82.83	**87.51**	**87.80**	85.08	83.77
Transferases	10	100	5	65.64	62.29	69.12	73.62	70.08	69.39
	25	100	5	65.19	**73.33**	72.60	**74.34**	69.97	67.39
Hydrolases	10	100	5	70.64	67.62	74.65	78.43	75.33	76.31
	25	100	5	69.88	73.86	**77.44**	**78.96**	75.38	74.41
Lyases	10	100	5	91.92	85.19	91.37	92.93	91.24	92.10
	25	100	5	91.30	90.72	**93.71**	**94.18**	91.65	91.26
Isomerases	10	100	5	91.92	86.52	91.43	93.43	91.55	92.62
	25	100	5	90.24	88.92	**93.57**	**94.25**	90.73	90.90
Ligases	10	100	5	90.11	84.54	85.77	90.56	89.52	91.15
	25	100	5	88.56	86.05	**89.83**	**91.56**	88.18	88.48

Table 4.11: F1 Scores for each classifier with different hyper-parameter sets of Word2vec model.

	W	N	l	***k*NN**	**SVM**	**Naive**	**RF**	**Ada Boost**	**ANN**
Oxidoreductases	10	100	5	0.82	0.83	0.84	0.86	0.84	0.84

	25	100	5	0.82	0.82	**0.87**	**0.88**	0.85	0.83
Transferases	10	100	5	0.61	0.57	0.67	0.72	0.68	0.67
	25	100	5	0.60	**0.72**	0.71	**0.73**	0.67	0.64
Hydrolases	10	100	5	0.68	0.64	0.73	0.78	0.74	0.75
	25	100	5	0.67	0.72	**0.77**	**0.78**	0.74	0.73
Lyases	10	100	5	0.92	0.85	0.91	0.93	0.91	0.92
	25	100	5	0.91	0.91	**0.94**	**0.94**	0.91	0.91
Isomerases	10	100	5	0.92	0.86	0.91	0.94	0.92	0.92
	25	100	5	0.90	0.89	**0.93**	**0.94**	0.91	0.91
Ligases	10	100	5	0.90	0.84	0.86	0.91	0.89	0.91
	25	100	5	0.88	0.86	**0.90**	**0.92**	0.88	0.88

Table 4.12: MCC Scores for each classifier with different hyper-parameter sets of Word2vec model.

	W	N	l	***k*NN**	**SVM**	**Naive**	**RF**	**Ada Boost**	**ANN**
Oxidoreductases	10	100	5	0.70	0.69	0.71	0.74	0.70	0.71
	25	100	5	0.70	0.70	**0.77**	**0.78**	0.73	0.71
Transferases	10	100	5	0.43	0.36	0.45	0.52	0.46	0.47
	25	100	5	0.42	**0.53**	0.52	**0.55**	0.49	0.45
Hydrolases	10	100	5	0.51	0.46	0.55	0.60	0.56	0.58
	25	100	5	0.50	0.55	**0.60**	**0.63**	0.57	0.56
Lyases	10	100	5	0.85	0.74	0.83	0.86	0.83	0.85
	25	100	5	0.84	0.83	**0.88**	**0.89**	0.84	0.83
Isomerases	10	100	5	0.85	0.76	0.83	0.87	0.84	0.86
	25	100	5	0.82	0.79	**0.87**	**0.89**	0.83	0.83
Ligases	10	100	5	0.82	0.72	0.73	0.82	0.80	0.83
	25	100	5	0.79	0.75	**0.81**	**0.84**	0.78	0.79

The classifiers perform under different parameters which are same with the parame-

ters in Section 4.1.2. Figure 4.9 shows the *k*NN classifier's F1 and MCC scores for different k values. Figure 4.10 and Figure 4.11 show the F1 and MCC scores for Random Forest and AdaBoost classifier for different number of estimators. According to *k*NN performance indicators, the lowest k value for the classifier performs better. For AdaBoost and Random Forest classifiers, the highest number of estimators performs better except a few cases such as the results of AdaBoost for the first and the second classes.

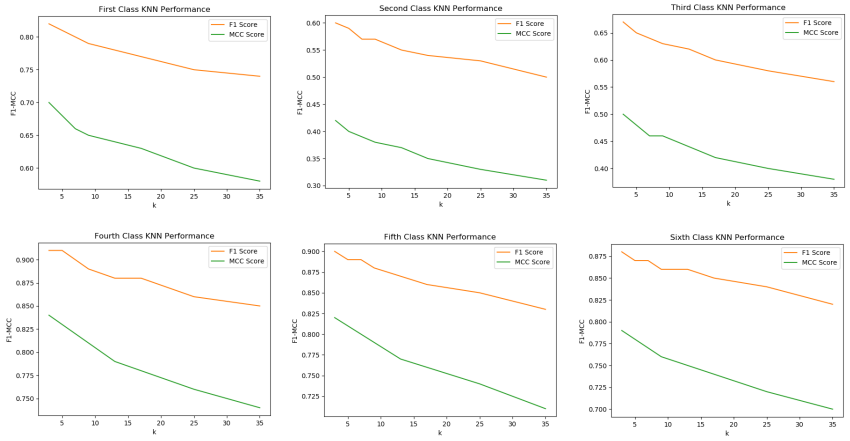

Figure 4.9: *k*NN performance scores with the k values.

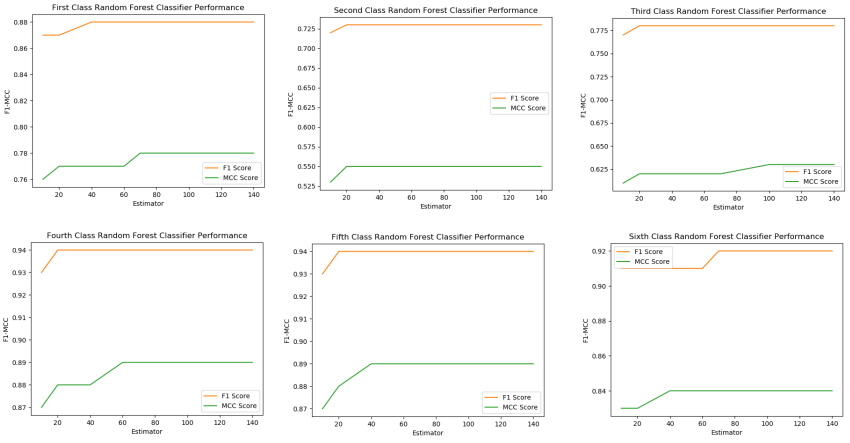

Figure 4.10: Random Forest classifier performance scores with different number of estimator values.

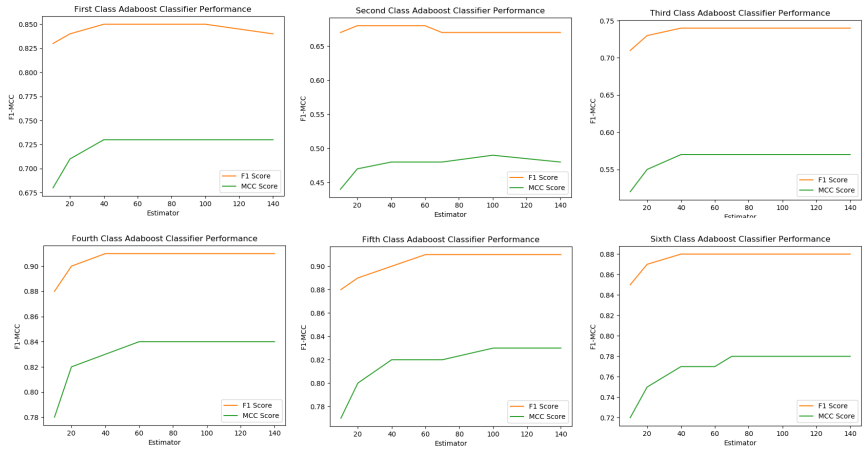

Figure 4.11: AdaBoost classifier performance scores with different number of estimator values.

Since the SVM classifier has more than one hyper-parameter, the performance indicators of SVM are shown in Table 4.13. The results indicate that one set of the C and the γ parameters results better than others, although there is an exception which is the results of the second class. The best hyper-parameters of SVM is 0.01 for C parameter and 1.00 for the γ parameter. The SVM classifier results from Section 4.1.2 indicate that performance scores are better with higher C values. Table 4.13 shows the highest C value and 1.00 for the γ parameter which are one of the best SVM hyper-parameters in Section 4.1.2 as good as the best hyper-parameter set of SVM in Section 4.1.2.

Table 4.13: MCC and F1 scores for different hyper-parameter values of SVM classifier.

Class	**kernel**	C	γ	**MCC Score**	**F1 Score**
Oxidoreductases	linear	1.00	auto	0.64	0.80
Oxidoreductases	poly	1.00	auto	0.42	0.60
Oxidoreductases	rbf	0.01	0.10	0.68	0.81
Oxidoreductases	rbf	0.01	1.00	**0.78**	**0.88**
Oxidoreductases	rbf	0.01	10.0	0.08	0.35
Oxidoreductases	rbf	1.00	0.10	0.70	0.82

Oxidoreductases	rbf	1.00	1.00	**0.75**	**0.86**
Oxidoreductases	rbf	1.00	10.0	0.61	0.78
Oxidoreductases	rbf	100.0	0.10	0.69	0.82
Oxidoreductases	rbf	100.0	1.00	0.75	0.86
Oxidoreductases	rbf	100.0	10.0	0.62	0.80
Oxidoreductases	sigmoid	1.00	auto	0.55	0.71
Transferases	linear	1.00	auto	0.48	0.7
Transferases	poly	1.00	auto	0.14	0.37
Transferases	rbf	0.01	0.10	0.41	0.60
Transferases	rbf	0.01	1.00	0.53	0.72
Transferases	rbf	0.01	10.0	0.02	0.34
Transferases	rbf	1.00	0.10	0.44	0.63
Transferases	rbf	1.00	1.00	0.49	0.67
Transferases	rbf	1.00	10.0	**0.54**	**0.76**
Transferases	rbf	100.0	0.10	0.43	0.62
Transferases	rbf	100.0	1.00	0.49	0.67
Transferases	rbf	100.0	10.0	**0.55**	**0.77**
Transferases	sigmoid	1.00	auto	0.39	0.58
Hydrolases	linear	1.00	auto	0.51	0.71
Hydrolases	poly	1.00	auto	0.25	0.46
Hydrolases	rbf	0.01	0.10	0.51	0.68
Hydrolases	rbf	0.01	1.00	0.61	0.77
Hydrolases	rbf	0.01	10.0	0.04	0.34
Hydrolases	rbf	1.00	0.10	0.55	0.72
Hydrolases	rbf	1.00	1.00	**0.59**	**0.75**
Hydrolases	rbf	1.00	10.0	0.54	0.76
Hydrolases	rbf	100.0	0.10	0.55	0.72
Hydrolases	rbf	100.0	1.00	**0.59**	**0.75**
Hydrolases	rbf	100.0	10.0	0.55	0.77
Hydrolases	sigmoid	1.00	auto	0.4	0.59
Lyases	linear	1.00	auto	0.76	0.87

Lyases	poly	1.00	auto	0.57	0.73
Lyases	rbf	0.01	0.10	0.83	0.90
Lyases	rbf	0.01	1.00	**0.88**	**0.94**
Lyases	rbf	0.01	10.0	0.13	0.37
Lyases	rbf	1.00	0.10	0.83	0.91
Lyases	rbf	1.00	1.00	**0.87**	**0.93**
Lyases	rbf	1.00	10.0	0.68	0.82
Lyases	rbf	100.0	0.10	0.82	0.90
Lyases	rbf	100.0	1.00	**0.87**	**0.93**
Lyases	rbf	100.0	10.0	0.70	0.83
Lyases	sigmoid	1.00	auto	0.70	0.82
Isomerases	linear	1.00	auto	0.75	0.87
Isomerases	poly	1.00	auto	0.50	0.66
Isomerases	rbf	0.01	0.10	0.80	0.89
Isomerases	rbf	0.01	1.00	**0.88**	**0.94**
Isomerases	rbf	0.01	10.0	0.03	0.34
Isomerases	rbf	1.00	0.10	0.79	0.89
Isomerases	rbf	1.00	1.00	**0.86**	**0.92**
Isomerases	rbf	1.00	10.0	0.71	0.84
Isomerases	rbf	100.0	0.10	0.79	0.88
Isomerases	rbf	100.0	1.00	**0.86**	**0.92**
Isomerases	rbf	100.0	10.0	0.72	0.84
Isomerases	sigmoid	1.00	auto	0.68	0.81
Ligases	linear	1.00	auto	0.75	0.87
Ligases	poly	1.00	auto	0.46	0.64
Ligases	rbf	0.01	0.10	0.73	0.84
Ligases	rbf	0.01	1.00	**0.82**	**0.90**
Ligases	rbf	0.01	10.0	0.04	0.34
Ligases	rbf	1.00	0.10	0.75	0.86
Ligases	rbf	1.00	1.00	**0.79**	**0.88**
Ligases	rbf	1.00	10.0	0.66	0.82

Ligases	rbf	100.0	0.10	0.74	0.85
Ligases	rbf	100.0	1.00	**0.79**	**0.89**
Ligases	rbf	100.0	10.0	0.68	0.83
Ligases	sigmoid	1.00	auto	0.68	0.81

The performance scores of classifiers are measured for hyper-parameters of these classifiers. The SVM classifier performs better for the enzyme and non-enzyme dataset, but for the first level classes, the Random Forest classifier performs better. In order to choose a successful classifier, the performance indicators for all datasets are compared. For the level 0 dataset, the difference between Random Forest and SVM classifier is relatively small. As a result, we select the Random Forest classifier to compute the overall results. Figure 4.12 show the F1 and MCC performance scores of Random Forest classifier for all datasets.

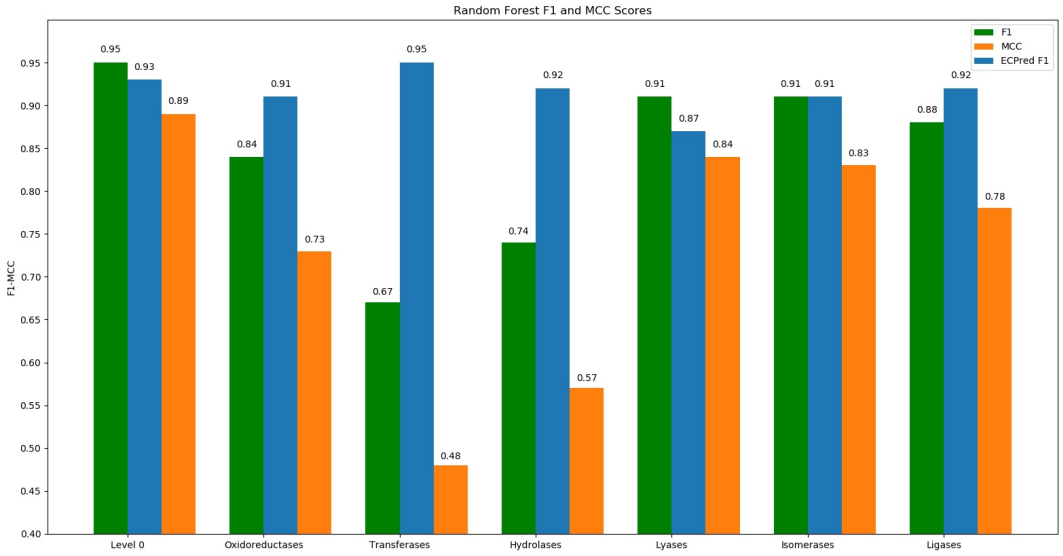

Figure 4.12: The plot of level one and level zero versus their Random Forest's F1 and MCC scores. Green color is used for F1 score and orange color is used for MCC score. Blue color is used for F1 score of ECPred [28] results.

The performance scores of the second class and the third class are lower than the average performance scores. The datasets are analyzed to find out the reason. First, the distribution of proteins in the negative datasets are given in Figure 4.13 where proteins

are evenly selected from other classes. Then, the average values of BLAST similarity scores between protein sequences in the positive datasets and negative datasets are computed by using BLAST. Figure 4.14 shows the heatmap for the similarity scores where the similarities between positive datasets and negative datasets of the second class and third class are nearly identical to the similarity scores of other classes. We couldn't find any reason for the low performance scores. The last step is visualizing the vector representations of data items in the second class by using the t-SNE algorithm. Figure 4.15 shows the distribution of vector representations of data items in 2D space where nearly half of the positive data items overlap with the negative data items and the distribution helps us to understand the low performance scores of the classifiers.

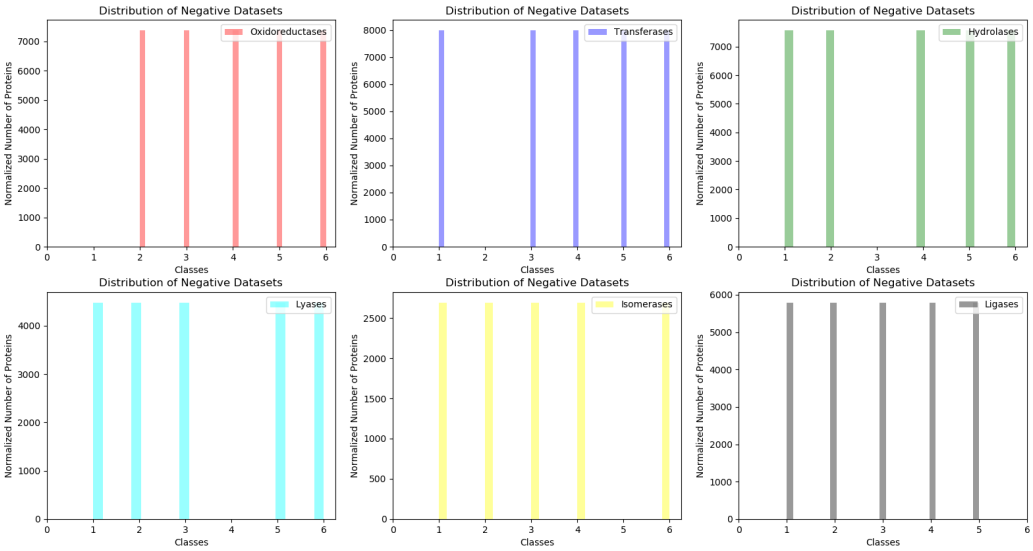

Figure 4.13: The plots show the distribution of protein sequences of negative datasets in other classes. The distributions indicate negative protein sequences are equally selected from other classes.

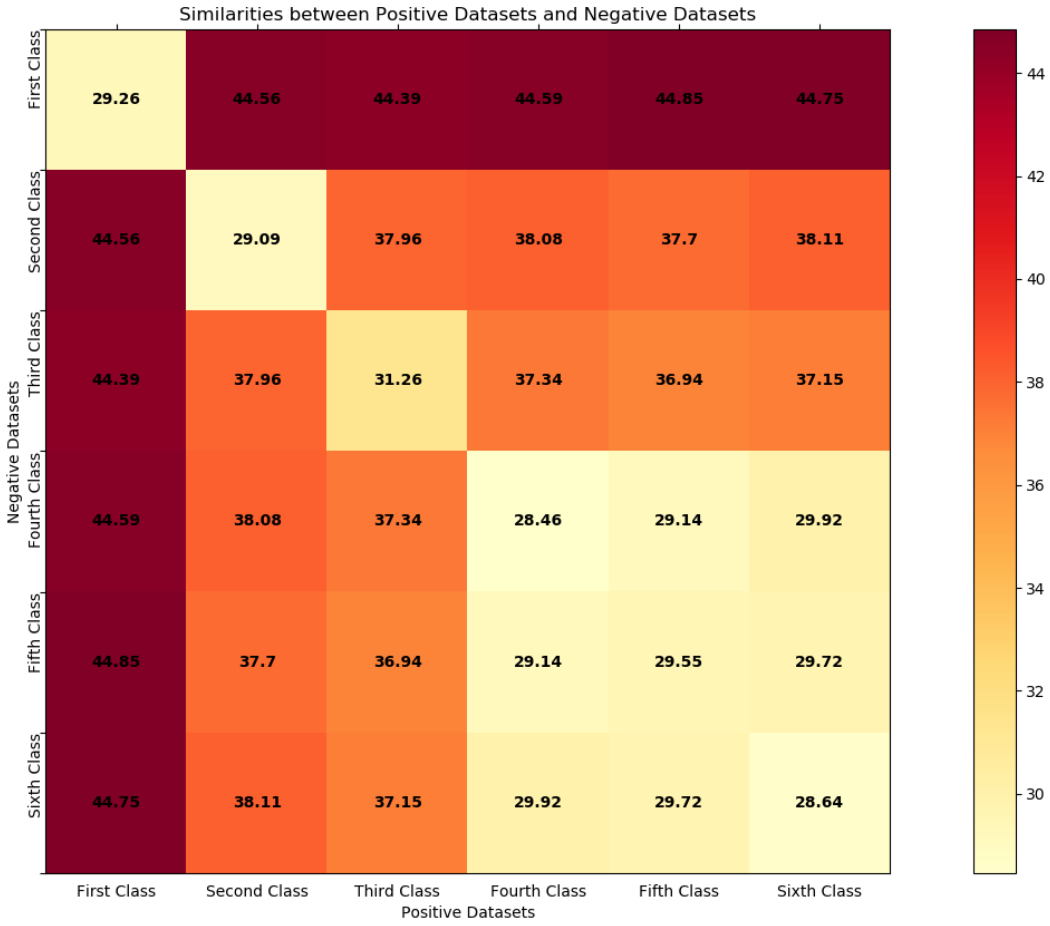

Figure 4.14: The heatmap for BLAST similarity scores between protein sequences in the positive datasets and negative datasets.

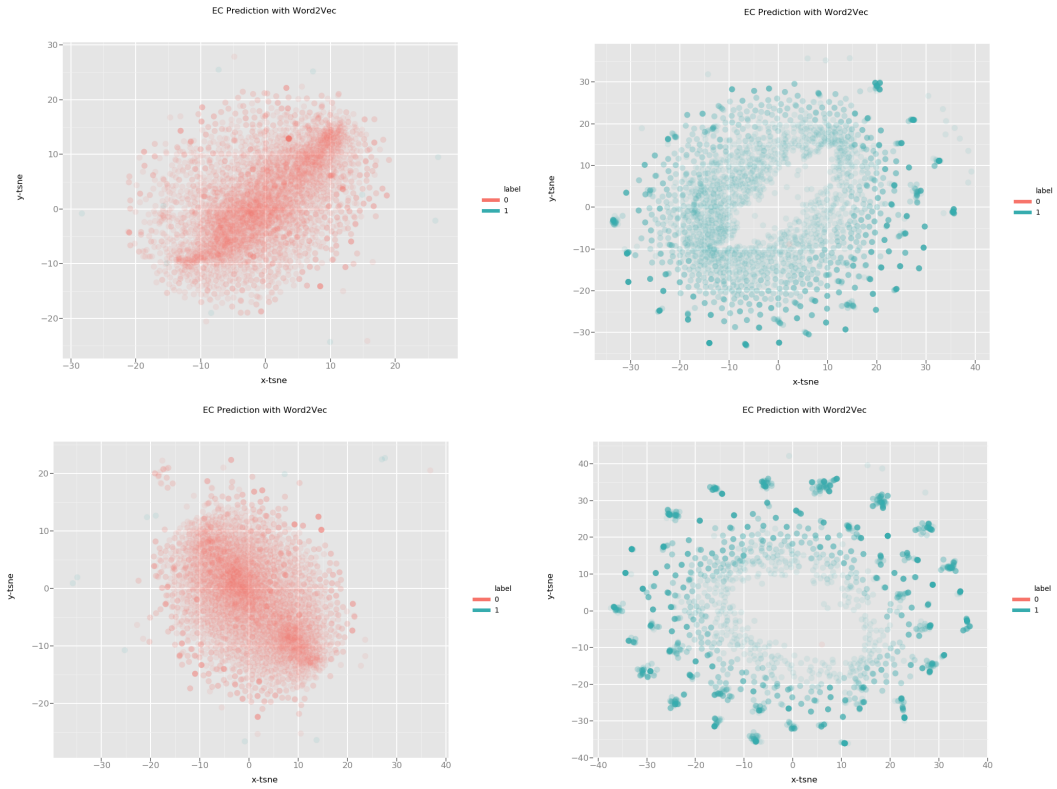

Figure 4.15: Vector representations of Word2vec models are visualized by using the t-SNE algorithm to project vectors onto 2D space. The first figure of second class belongs to the vector representations of the negative data items in the second class and the second figure of second class belongs to the vector representations of the positive data items in the second class. In order to compare the classes, vector representations of the fourth class are given at the second row of figures.

The training time of Word2vec model and classifiers is measured with one of the hyper-parameter set where N is 100, W is 10, l is 5 and vector calculation for a sequence is averaging method. Figure 4.16 shows the training time of first class of EC numbers in minutes against the number of protein sequences in the training dataset.

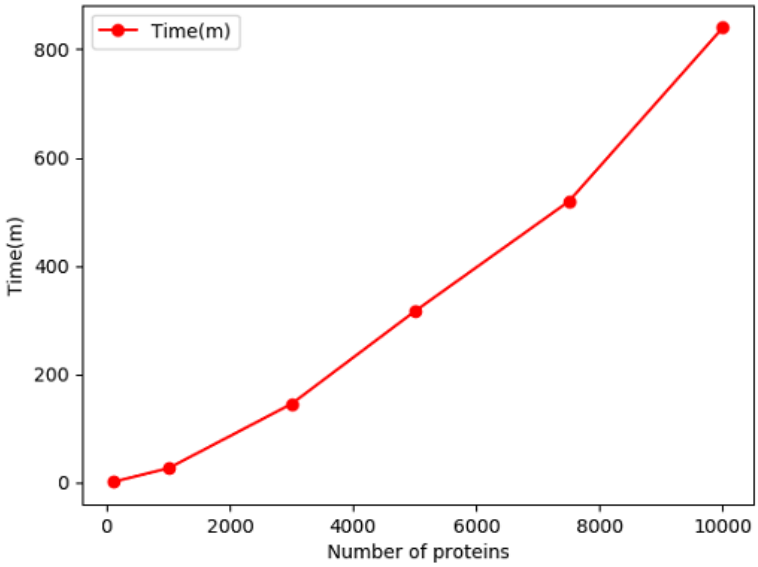

Figure 4.16: The plot of training time of Word2vec model and classifiers against the number of proteins in the training dataset.

4.1.4 Visualization of EC Numbers

Word2vec model represents protein sequences in the vector space which is used to predict enzyme commission classes. The visualization tool aims to visualize the result of EC Number predictions which are obtained by using the word embedding approach. This visualization tool works browser based and presents a new approach to demonstrate the Enzyme Commission Number as a tree. The visualization tool is a browser-based application which requires just a web browser without any additional dependencies. To the best of our knowledge, it is the first time the enzyme commission number is visualized in a web-based tool. Also, end-user has an interface to upload files of predictions which could be useful to compare the results of predictions. Options of vis.js [36] is used for the Visualization tool to achieve star topology tree. Star topology uses the central node which is named as the root node. Prediction results include protein id, hierarchical prediction of protein and confidence score for each class in the hierarchical structure. Figure 4.17 is the screenshot of the visualized prediction results of EC Classes. There are 859 nodes in the EC tree including one

root, 6 main, 55 sub-family, 163 sub-sub-family, and 634 substrate numbers. The root node has child nodes which are the 6 main classes of EC number. Since it isn't efficient to show all nodes in one tree, only requested nodes are shown in the tree. All EC number trees include root node in order to give an initial point to expand the tree. Double click on any node expanse the child nodes. If there isn't any child node of double clicked node, depth-first search is used to expand grandchild nodes. Only one level of nodes is expanded as a result of the user's action.

Non-enzyme and non-predicted proteins have information which is presented as a modal view at the right bottom of the web page. Since the non-enzyme and non-predicted proteins are irrelevant for some of end-users, the model can be hidden with a single button.

In order to stress the distribution of prediction results, graph coloring is implemented. In order to assign a color to a node, the number of child nodes is calculated. According to the number of child nodes, a color is assigned. A darker color indicates a higher number of child nodes. In order to distinguish the colors of the main classes, each main class is assigned to a different color scheme. These color schemes are similar to map color schemes which are easy to distinguish between each other. For instance, the fifth class which is named as isomerases uses a tone of blue.

- Scheme of orange is assigned to first class which is named as oxidoreductases.
- Scheme of green is assigned to second class which is named as transferases.
- Another scheme of green is assigned to third class which is named as hydrolases.
- Scheme of blue is assigned to fourth class which is named as lyases.
- Scheme of blue is assigned to fifth class which is named as isomerases. This scheme is darker than the scheme which is used by fourth class.
- Scheme of purple is assigned to sixth class which is named as ligases.

A node is associated with its own information and child nodes information. Also, information about the child nodes which isn't fully connected with an associated node

is presented. For instance, prediction output contains results of the sixth main class and sub-sub-classes of the sixth main class, but there isn't any result of subclasses of the sixth main class. In this case, information about sub-sub-classes is associated with the node of the sixth main class. Associated enzyme information represented at the top of the web page with the interaction of end-user. Since the number of associated enzymes can be varied for different result sets, a web slider is used to show the information area. Each information is framed to separated from other information boxes. Each information box contains a protein id, prediction score. Since there can be too many information boxes, information about single clicked node is shown.

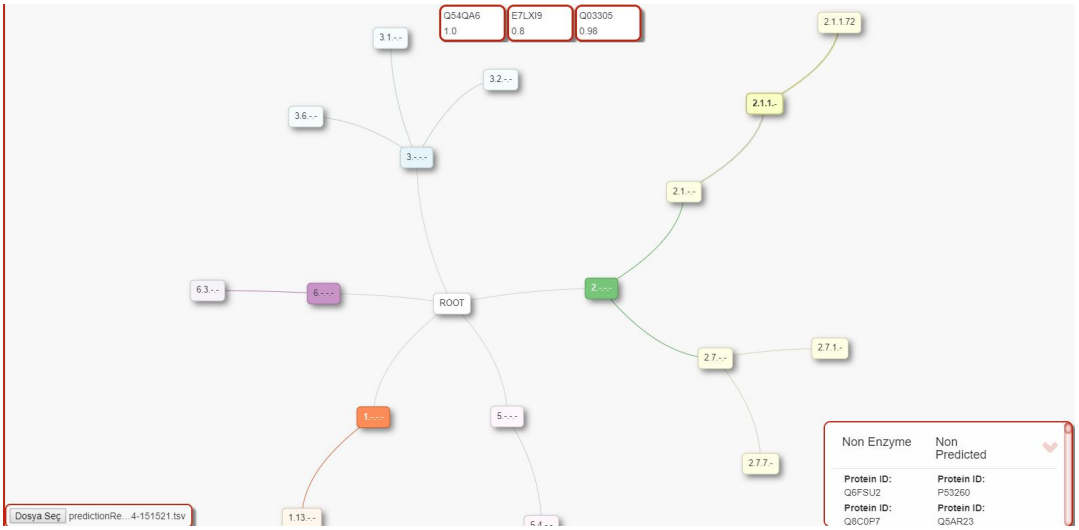

Figure 4.17: Screenshot of the predicted EC classes.

4.2 Discussion

In this section, the results of Word2vec models and classifiers are inspected. The similarities of protein sequences are computed with the feature vectors of protein sequences which are obtained from Word2vec models. The similarities are compared with the BLAST similarity scores to optimize the hyper-parameters of the Word2vec and the calculation methods of feature vectors of the protein sequences. We use the enzyme and non-enzyme dataset to compare Word2vec models with BLAST. Deviation values of 1000 most similar proteins and 1000 most dissimilar proteins are calculated and the Pearson correlation of the similar and dissimilar proteins is cal-

culated. Deviation results are compared and two of the hyper-parameter sets have two smallest deviation values. We perform 5-fold cross-validation with classifiers for all hyper-parameter sets to confirm the deviation results. The same two hyper-parameter sets perform better than others. As a result, these two hyper-parameter sets are selected and the experiments on the first level datasets are conducted with these hyper-parameter sets. The results which are obtained from the experiments on the first level dataset show that one of the hyper-parameter sets is better than others. Random Forest classifier is chosen since the performance scores of Random Forest classifier are better for most of the results. Vector representations of the second class and the third class of EC classes are classified and the classification results of these classes are worst than other classes since the vector representations of positive proteins are similar to the negative proteins in these datasets.

The F1 scores of ECPred [28] are shown in Figure 4.12. The average F1 score of ECPred is 0.93 but the dataset of this study is filtered by using UniRef. In this study, only the level 0 dataset is the same dataset which is used by ECPred. Sub-family classes contain fewer protein sequences which are more similar than the protein sequences of the main classes. Since the protein sequences are more similar for sub-family classes, we expect better classification results for substrate classes.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this study, biological sequences are represented as vectors by using Word2vec skip-gram model. Prediction of protein functions from protein sequences by using vector representations and classifiers is the main objective of this paper. First, biological sequences are split into equal length subsequences which are treated as words. Word2vec model is trained with the subsequences to represent subsequences as vectors which are the hidden node values of the Word2vec model. In order to train the Word2vec model, we use several sets of hyper-parameters and calculation method of feature vectors of sequences from the real-valued vectors of subsequences. In order to optimize the hyper-parameter set and the calculation method, similarities between vectors of sequences are calculated and compared with BLAST similarity scores. According to similarities between the vector representation of Word2vec and BLAST scores, we select the optimal hyper-parameter set and the calculation method.

The high-dimensional vector representation of sequences is projected into 2D space with using t-SNE algorithm which is used to visualize the distribution of high dimensional vectors. The data items are classified with several classifiers such as *k*NN, SVM, Naive Bayes, and Random Forest to identify the functionalities of biological sequences. We apply cross-validation on the datasets to optimize the hyper-parameter sets of the classifiers. Also, classifier performances are evaluated for different hyper-parameter sets of Word2vec. The comparison between classifiers shows that SVM and Random Forest classifiers are performed better for the vector representations. Since the Random Forest classifier's performance scores are better for most of the datasets, we show the performance metrics of Random Forest classifier in Figure 4.12. The F1 score value for the level 0 dataset is 0.97 and Matthews correlation

coefficient score obtained as 0.93. The average F1 score value for EC main classes obtained as 0.87 and 0.70 for MCC.

A web-based interactive visualization tool is developed to trace the relations between predicted enzymes. Enzyme Commission Number is visualized as a tree and predicted enzymes are associated with the nodes of the EC Number tree. Also, non-predicted and non-enzyme proteins are shown on the web page. Since there can be too many predictions, a graph coloring mechanism is implemented to stress the distribution of predicted enzymes.

In the future, training Word2vec models for all of the EC classes and evaluating the performances of the classifiers are planned. The execution time of training the Word2vec model is the major issue that prevents us from training all of the EC classes. There are 858 EC classes which have 30 or more protein annotations and these EC classes are 6 main, 55 subfamily, 163 sub-subfamily, and 634 substrate classes. Since the training time nearly takes 15 hours for the first class, training Word2vec models for all of the EC classes requires enormous time. We classify data items with binary classification, each EC class is trained and classified individually. In the future, we plan to classify each level of EC classes with a multiclass classifier. Also, experiments on other topics and datasets can be evaluated such as DNA sequence classification. The web-based visualization tool will be available to the potential users and different prediction results will be visualized such as GO annotations.

REFERENCES

[1] The Gene Ontology Consortium, "The Gene Ontology Resource: 20 years and still GOing strong," *Nucleic Acids Research*, vol. 47, pp. D330–D338, 11 2018.

[2] M. D. Yandell and W. H. Majoros, "Genomics and natural language processing," *Nature Reviews Genetics*, vol. 3, no. 8, pp. 601–610, 2002.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," pp. 1–12, 2013.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Altschul_1990_5424.pdf," 1990.

[5] D. Kimothi, A. Soni, P. Biyani, and J. M. Hogan, "Distributed representations for biological sequence analysis," *CoRR*, vol. abs/1608.05949, 2016.

[6] E. Asgari and M. R. K. Mofrad, "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics," *PLOS ONE*, vol. 10, p. e0141287, nov 2015.

[7] O. CNX, "Openstax, biology." `http://cnx.org/contents/185cbf87-c72e-48f5-b51e-f14f21b5eabd@9.85`, May 2015. [Online; accessed 4-August-2019].

[8] P. Romero, "Bioinformatics: Sequence and Genome Analysis," *Briefings in Bioinformatics*, vol. 5, no. 4, pp. 393–396, 2006.

[9] V. Atalay, "Representing Protein Sequences With Bag-Of-Features Model For Pattern Recognition and Machine Learning," 2019.

[10] S. B. Needleman and C. D. Wunsch, "A general method applicable to search for similarities in amino acid sequence of 2 proteins," *Journal of molecular biology*, vol. 48, pp. 443–53, 04 1970.

[11] T. Smith and M. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195 – 197, 1981.

[12] B. Tay, J. K. Hyun, and S. Oh, "A machine learning approach for specification of spinal cord injuries using fractional anisotropy values obtained from diffusion tensor images," *Computational and mathematical methods in medicine*, vol. 2014, p. 276589, 01 2014.

[13] M. Firman Aji Saputra, T. Widiyaningtyas, and A. Wibawa, "Illiteracy classification using k means-naïve bayes algorithm," *JOIV : International Journal on Informatics Visualization*, vol. 2, p. 153, 05 2018.

[14] R.Gandhi, "Support vector machine - introduction to machine learning algorithms." `https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47`, June 2018. [Online; accessed 4-August-2019].

[15] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," pp. 144–152, 2004.

[16] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[17] T. Ho, "Random decision forests," vol. 1, pp. 278 – 282 vol.1, 09 1995.

[18] N. Kumar, "Random forest algorithm, an interactive discussion." `https://www.linkedin.com/pulse/random-forest-algorithm-interactive-discussion-niraj-kumar/`, June 2016. [Online; accessed 4-August-2019].

[19] S. Gupta, "A short introduction to heavy-ion physics," vol. 14, no. 5, pp. 771–780, 2015.

[20] Wikipedia contributors, "Artificial neural network — Wikipedia, the free encyclopedia," 2019. [Online; accessed 4-August-2019].

[21] T. Kozai and G. Niu, "DeepLearningbook," *Plant Factory: An Indoor Vertical Farming System for Efficient Quality Food Production*, pp. 3–5, 2015.

[22] Laurens van der Maaten and G. Hinton, "Visualizing Data using t-SNE Laurens," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[23] G. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in Neural Information Processing Systems 15*, pp. 833–840, 2003.

[24] P. Ng, "dna2vec: Consistent vector representations of variable-length k-mers," pp. 1–10, 2017.

[25] H. Öztürk, E. Ozkirimli, and A. Özgür, "A novel methodology on distributed representations of proteins using their interacting ligands," *Bioinformatics*, vol. 34, no. 13, pp. i295–i303, 2018.

[26] T. Wittkop, D. Emig, S. Lange, S. Rahmann, M. Albrecht, J. H. Morris, S. Böcker, J. Stoye, and J. Baumbach, "Partitioning biological data with transitivity clustering," *Nature methods*, vol. 7, p. 419—420, June 2010.

[27] A. Enright, S. Van Dongen, and C. Ouzounis, "An efficient algorithm for large-scale detection of protein families," *Nucleic acids research*, vol. 30, pp. 1575–84, 05 2002.

[28] A. Dalkıran, "PREDICTION OF ENZYMATIC PROPERTIES OF PROTEIN SEQUENCES," no. 19, p. 74, 2017.

[29] B. E. Suzek, Y. Wang, H. Huang, P. B. McGarvey, C. H. Wu, and U. Consortium, "Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches," *Bioinformatics*, vol. 31, no. 6, pp. 926–932, 2014.

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,

D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

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

[33] H. Wickham, *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[34] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[35] C. Camacho, G. Coulouris, V. Avagyan, N. Ma, and J. Papadopoulos, "Blast+: architecture and applications," *BMC Bioinform.*, vol. 10, pp. 1–9, 01 2009.

[36] B. Almende, "Vis.js." http://visjs.org/, 2015. [Online; accessed 4-August-2019].