Math 499 Capstone project

Protein Family Classification using embedding methods

Damilya Saduakhas

Supervisor: Zhenisbek Assylbekov
Second reader: Rustem Takhanov

Nazarbayev University
April 2020

Contents

Abstract

This capstone project examines the performance of existing embedding based alignment-free methods for protein family classification tasks. The distributed continuous representation of biological sequences such as DNA and proteins can be analyzed using algorithms that are based upon Natural Language Processing models such as Word2Vec. The performance of ProtVec proposed by Asgari et.al. (2015) was analyzed and compared to its further improvements and the opportunities of embedding based methods in classification tasks were discussed. The data were obtained from the Swiss-Prot database, and 324,018 manually annotated protein sequences were used for protein family classification task of 7,027 families. This paper will test different advantages and will try to explain the motivation behind using the embedding methods for classification, despite the existence of advanced alignment methods with high accuracy. The further modifications to change the metrics to non-Euclidean ones and the use of hybrid models were proposed.

1 Introduction and Background

The language of our natural world consists of different types of biological sequences such as DNA and proteins. It is a language that our nature speaks the same way as people communicate in English. The importance that these biological units play in moving forward the science borders is immense. This thesis will focus particularly on protein sequences, even though a generalization of further discussed algorithms to a general type of biological sequences can be made with ease. The particular task considered given a variety of existing issues is a protein family classification problem. Thanks to Next generation sequencing (NGS) which has revolutionized the world of biological sequences, the amount of data available for bioinformatics and data analysis (NGS is a technology that allows in one experiment to obtain the sequence of DNA molecules) has increased tremendously.

Determination of the family of protein and its function is usually done in a wet laboratory experiment; however, it can be expensive and time-consuming to perform. Therefore, the need for quick classification methods gave rise to a field of bioinformatics, where one can apply computational methods. The protein sequence consists of basic units - amino acids, overall there are 20 types each assigned with a capital letter. The sequence of capital letters is a typical representation of a protein sequence (See Figure 1), which allows us to consider a sequence equivalent to a sentence of words and apply Natural Language Processing algorithms.

Why do we need to classify protein in families? The main motivation comes from the heredity since proteins in one protein family usually inherit some biophysical and biochemical properties from ancestor proteins.

Currently, proteins can be classified into different groups according to:

- families, they belong to
- domains they have
- sequence features [1]

Protein family is a group of proteins that share a common evolutionary origin, reflected by their related functions and similarities in sequence or structure. The evolution of proteins led to having same features inherited from a common ancestor, which allows creating protein families in hierarchical forms according to the number of features shared [2]. There are also superfamilies and subfamilies existing, however, for generalizations, we will simply take the families as mentioned in the Swiss-Prot database to replicate the results of the ProtVec model's accuracy and attempt to apply it to other protein databases too. The protein database grows every year following the development of protein purification techniques, increasing the need for fast classification tools. Classifying the protein simply using its sequence representation is an easy-to-apply tool, which could be used for a pre-analysis stage to identify the functions of unknown protein not available in common databases.

G	Glycine	Gly	**P**	Proline	Pro
A	Alanine	Ala	**V**	Valine	Val
L	Leucine	Leu	**I**	Isoleucine	Ile
M	Methionine	Met	**C**	Cysteine	Cys
F	Phenylalanine	Phe	**Y**	Tyrosine	Tyr
W	Tryptophan	Trp	**H**	Histidine	His
K	Lysine	Lys	**R**	Arginine	Arg
Q	Glutamine	Gln	**N**	Asparagine	Asn
E	Glutamic Acid	Glu	**D**	Aspartic Acid	Asp
S	Serine	Ser	**T**	Threonine	Thr

Figure 1: Amino acids encoded with one and three letters

Thus, the bioinformatics field tried to adapt existing NLP methodologies that dealt with spoken languages and apply it to the language of nature, to biological sequences. In this capstone project, the application of concrete NLP models adapted to biological sequences will be discussed. Even though the Word2Vec model is not a state-of-the-art algorithm of NLP, in this thesis the advantages of this method applied to bio-sequences will be shown and disadvantages of some more advanced models tailored to assigned tasks as protein family classification will be pointed out.

The presented biological embeddings will be primarily tested using protein family classification tasks trained on the Swiss-Prot database, which includes only manually reviewed protein sequences, therefore assuring us in the correctness of assigned families. Generally, obtained biological embeddings can also be used further and can be analyzed using the biological properties such as Van der Wal volume, hydrophobicity, etc.

A typical protein sequence is represented as the continuous string of letters, where each letter stands for the amino acid as stated in Figure 1, and the length of the protein sequence is usually determined by its function. The following sequence is a typical representation of the protein sequence to be analyzed.

```
MAFSAEDVLKEYDRRRRMEALLLSLYYPNDRKLLDYKEWSPPRVQVECPKAPVEWNNPPS
EKGLIVGHFSGIKYKGEKAQASEVDVNKMCCWVSKFKDAMRRYQGIQTCKIPGKVLSDLD
AKIKAYNLTVEGVEGFVRYSRVTKQHVAAFLKELRHSKQYENVNLIHYILTDKRVDIQHL
EKDLVKDFKALVESAHRMRQGHMINVKYILYQLLKKHGHGPDGPDILTVKTGSKGVLYDD
SFRKIYTDLGWKFTPL
```

Figure 2: Example of protein sequence - Putative transcription factor 001R

An analogy between the Natural language and language of life can be drawn as following:

Natural language	Protein
word	3-gram
sentence	sequence
Corpus (e.g. book)	Protein Database (e.g. Swiss-Prot)

2 Methods

Natural Language Processing (NLP) is a field that uses various techniques from the fields like computer science, linguistics and artificial intelligence to help computers understand human language. The goal of NLP is to derive valuable and interpretable information from the analysis of human languages.

2.1 Word2Vec

The digital representation of words is required to work with them in a digital system, which can only understand 0s and 1s. Therefore, there have been many attempts done previously to create such a system that would preserve the relations between words and will also be manageable for further applications like text analysis, classification, syntax analysis, etc.

The first attempt to use word embedding in language modeling tasks was proposed back in 2001 by Bengio et al. and was initially called as learning distributed representation for each word [3]. Later in 2008, Ronan and Jason developed a concept of "pre-trained embeddings" that can be used for immediate application. Both word embeddings and pre-trained embeddings found their realization in Word2Vec that was popularized in 2013 [4], [5]. The "Word2Vec" was developed by Mikolov and his team from Google. It is a software package consisting of 2 algorithms — continuous bag of words (CBOW) and continuous Skip-gram, as well as 2 training methods — negative sampling and hierarchical softmax. The paper [6] discussed the advantages of proposed methods compared to the performance of the Feed-forward Neural Net Language Model (NNLM) and Recurrent Neural Net Language Model (RNNLM) on the syntactic and semantic tasks associated with language. This particular paper only introduced the first part of Word2Vec, the two algorithms, and their brief method of work.

The second follow-up paper 'Distributed Representations of Words and Phrases and their Compositionality' (2013) explained implemented methods of unsupervised learning based on the distributional hypothesis in greater detail[4]. According to the hypothesis, the words located in the same context tend to have a similar meaning. Following this hypothesis, the distributed representation of words allows one to use cosine similarity to

check the obtained word embeddings' quality. For vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ the cosine similarity is

$$\frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \in [-1, 1] \tag{1}$$

Analogously, if the words are represented as vectors in n-dimensional space, the cosine similarity would describe the cosine of an angle between any two vectors in that space, i.e. closeness. The comparison tables presented in the paper show that using a simple unsupervised shallow neural network allows one to obtain quality word embeddings, which can be later used for various Natural Language Processing (NLP) tasks, such as SemEval-2012 Task 2 or Microsoft Research Sentence Completion Challenge discussed in the paper. The comparison of different model architectures showed that CBOW architecture works better on the syntactic tasks or at least comparable to other models, and Skip-gram works best on semantic tasks. This paper also gave some suggestions regarding how to reduce the complexity of the neural network, one solution was using hierarchical softmax and application of log-linear models.

They showed the mathematical formulation of the objective function which is the maximization of the average log probability defined by softmax function. To efficiently estimate the softmax function, they introduced the hierarchical softmax which allows computing log_2(W) nodes instead of W nodes. However, later authors argue that instead of using hierarchical softmax they will use the alternative Noise Contrastive Estimation (NCE) and simplify it further to obtain Negative Sampling (NEG). NEG is used to decrease the computational cost and increase the efficiency of the model. The concrete formulas for the NEG and hierarchical softmax are introduced and explained in greater detail in following sections.

2.1.1 Skip-Gram

Since ProtVec is a model based on the skip-gram model with negative sampling, these two methods will be discussed in detail to understand the main concept of ProtVec and build on it with further advancements.

In the skip-gram model, each word is represented as a pair of d - dimensional vectors, input and output vectors, which are then used in conditional probability. We assume that the word is indexed as i in the dictionary, and its vectors are represented as $v_i \in \mathbb{R}^d$ when it is the central target word, and $u_i \in \mathbb{R}^d$ when it is a context word. Let the central target word w_c and context word w_o be indexed as c and o respectively in the dictionary. According to [4] the conditional probability of generating the context word for the given central target word can be obtained by performing a softmax operation on the vector inner product:

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}, \tag{2}$$

w_o - context word with index o in the vocabulary,
w_c - central target word with index c in the vocabulary,
$\mathbf{v}_i \in \mathbb{R}^d$ - is d-dimensional vector for the word i when it is a central target word,
$\mathbf{u}_i \in \mathbb{R}^d$- is d-dimensional vector for the word i when it is a context word,
$\mathcal{V} = \{0, 1, \ldots, |\mathcal{V}| - 1\}$ - vocabulary set.

Mathematically speaking the original objective function of skip-gram, is to maximize the following average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c,\ j\neq 0}\log P(w^{(t+j)}\mid w^{(t)}). \tag{3}$$

c - the size of the training context,
$w^{(t)}$ - center word from a sequence of training words $w^{(1)}, w^{(2)}...w^{(T)}$,
$w^{(t+j)}$ - (t+j)th word in a training sequence,
T - the size of training sequence.

Deriving equation (3) using this notation, first we will obtain the likelihood function:

$$\prod_{t=1}^{T}\prod_{-c\leq j\leq c,\ j\neq 0}P(w^{(t+j)}\mid w^{(t)}) \tag{4}$$

and taking logarithm of and averaging it will result in equation (3).

While training the skip-gram model, we are going to learn the model parameters, i.e. the central target word vector and context word vector for each word, by using maximum likelihood estimation, where we simply maximize the likelihood function.

But the equation (2) will not be used for further calculations, since during optimization stage the $\nabla\log P(w_o|w_c)$ will require number of operations proportional to the $\mathcal{V}$, size of vocabulary:

$$\log P(w_o\mid w_c)=\mathbf{u}_o^\top\mathbf{v}_c-\log\left(\sum_{i\in\mathcal{V}}\exp(\mathbf{u}_i^\top\mathbf{v}_c)\right). \tag{5}$$

Differentiating with respect to $\mathbf{v}_c$:

$$\begin{aligned}\frac{\partial\log P(w_o\mid w_c)}{\partial\mathbf{v}_c}&=\mathbf{u}_o-\frac{\sum_{j\in\mathcal{V}}\exp(\mathbf{u}_j^\top\mathbf{v}_c)\mathbf{u}_j}{\sum_{i\in\mathcal{V}}\exp(\mathbf{u}_i^\top\mathbf{v}_c)}\\&=\mathbf{u}_o-\sum_{j\in\mathcal{V}}\left(\frac{\exp(\mathbf{u}_j^\top\mathbf{v}_c)}{\sum_{i\in\mathcal{V}}\exp(\mathbf{u}_i^\top\mathbf{v}_c)}\right)\mathbf{u}_j\\&=\mathbf{u}_o-\sum_{j\in\mathcal{V}}P(w_j\mid w_c)\mathbf{u}_j.\end{aligned} \tag{6}$$

Thus, we can see on the single example of partial derivative with respect to the input vector v_c, that the computation will require the summation of probabilities from the whole vocabulary, making it rather complex.

2.1.2 Negative Sampling and Hierarchical Softmax

Negative sampling addresses the issue of updating too many weights in the neural network. Even though it is based on Skip-Gram, it is, in essence, an optimization of the model by

changing each training sample only a small percentage of the weights, rather than updating all of them each iteration. While our corpus of text can range anything from a few thousand up to a million sequences, it can take rather long time training even such shallow network [7]. The negative sampling also modifies the original objective function. Now, the objective function will compute given a context window for the central target word w_t , we will treat it as an event for context word w_{t+j} to appear in the context window and compute the probability:

$$P(D = 1 \mid w_{t+j}, w_t) = \sigma({v'_{w_{t+j}}}^{\top} v_{w_t}), \tag{7}$$

The model in the current state will include only 'positive examples', however, if we try to maximize the joint probability to 1, it will be possible only when all the word vectors are equal and their values approach infinity. Vectors in that form are useless, thereby urging the need for 'negative examples'. They will be introduced as noise words that didn't occur in the context of the central target word during training and will be randomly chosen from the corpus using particular distribution.

We will first consider training the word vector by maximizing the joint probability of all events in the text sequence using only positive examples. Given a text sequence of length T, we assume that the word at time step t is $w^{(t)}$ and the context window size is c. Now we consider maximizing the joint probability:

$$\prod_{t=1}^{T} \prod_{-c \le j \le c,\ j \ne 0} P(w^{(t+j)} \mid w^{(t)}), \tag{8}$$

we will approximate the conditional probability as

$$P(w^{(t+j)} \mid w^{(t)}) = P(D = 1 \mid w^{(t)}, w^{(t+j)}) \prod_{k=1,\ w_k \sim P(w)}^{K} P(D = 0 \mid w^{(t)}, w_k). \tag{9}$$

where w_k has distribution $P(w)$ and $k = 1, \ldots, K$.

Negating the conditional log probability will give the logarithmic loss:

$$\begin{aligned} -\log P(w^{(t+j)} \mid w^{(t)}) &= -\log P(D = 1 \mid w^{(t)}, w^{(t+j)}) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log P(D = 0 \mid w^{(t)}, w_k) \\ &= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^{\top} \mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log\left(1 - \sigma\left(\mathbf{u}_{h_k}^{\top} \mathbf{v}_{i_t}\right)\right) \\ &= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^{\top} \mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log \sigma\left(-\mathbf{u}_{h_k}^{\top} \mathbf{v}_{i_t}\right) \end{aligned} \tag{10}$$

Compared to the previous calculation of gradient when the objective function used only positive examples, the complexity was proportional to the vocabulary size, however now, the gradient is linearly dependent on constant K, which reduces the computational cost.

Selecting negative samples is done using the unigram distribution raised to the power 3/4:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} \left(f(w_j)^{3/4} \right)} \tag{11}$$

order of words in the history does not influence the projection.

3 ProtVec

In 2015 Asgari and Mofrad, created a model that is based on the Word2Vec algorithm to create specific embeddings - biological [8]. Following similarity between a biological sequence and a sentence, they used a typical method of breaking the whole sequence into k-mers to obtain "words". Setting the k=3, they break a protein sequence into non-overlapping trimers with a sliding window equal to 1, as seen in Figure 3:

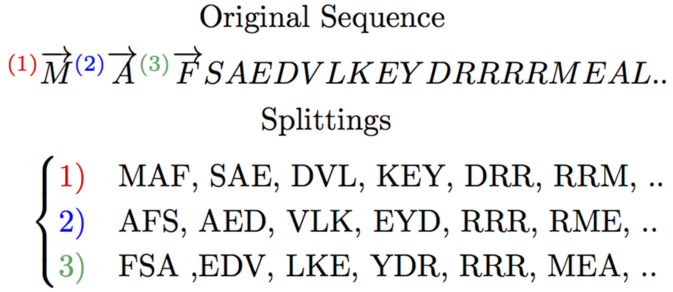


Figure 3: Split of a protein sequence.
Each sequence is represented through 3 sequences with shifting window size=1 and each divided into a triplet of amino acids. Source: Asgari and Mofrad, 2015 [8]

The main advantage is that such embeddings need to be trained only once and can be used immediately for further application. However, the main disadvantage is that this method is insensitive to order. For example, if the sequence "MAF-SAE-DVL" and different sequences "DVL-MAF-SAE" to be met in the training data, both will have the same vector representation, since the sum of trimers is used for the representation of whole sequences. This insensitiveness may cause the inaccuracy of representation of the properties of such sequences since both potentially can be part of different protein families

The data that was used in the paper of Asgari et al. was from a manually annotated database Swiss-prot, which included 324,018 protein sequences at the time of publication and they belonged to 7,027 protein families. They claimed to have obtained an average accuracy of classifying protein families $93\% \pm 0.06\%$. However, during analysis, the following discrepancy was found. In the original data provided by Asgari et al. we have found 7027 FamilyIDs but only 6967 FamilyDescription, meaning that some protein families had formed subfamilies of proteins within themselves. This discrepancy was also mentioned in [9]

	SwissProtAccessionID	LongID	ProteinName	FamilyID	FamilyDescription	Sequence
0	Q6GZX4	001R_FRG3G	Putative transcription factor 001R	Pox_VLTF3	Poxvirus Late Transcription Factor VLTF3 like	MAFSAEDVLKEYDRRRRMEALLLSLYYPNDRKLLDYKEWSPPRVQV...
1	Q6GZX3	002L_FRG3G	Uncharacterized protein 002L	DUF230	Poxvirus proteins of unknown function	MSIIGATRLQNDKSDTYSAGPCYAGGCSAFTPRGTCGKDWDLGEQT...
2	Q6GZX0	005R_FRG3G	Uncharacterized protein 005R	US22	US22 like	MQNPLPEVMSPEHDKRTTTPMSKEANKFIRELDKKPGDLAVVSDFV...
3	Q91G88	006L_IIV6	Putative KilA-N domain-containing protein 006L	DUF3627	Protein of unknown function (DUF3627)	MDSLNEVCYEQIKGTFYKGLFGDFPLIVDKKTGCFNATKLCVLGGK...
4	Q197F3	007R_IIV3	Uncharacterized protein 007R	DUF2738	Protein of unknown function (DUF2738)	MEAKNITIDNTTYNFFKFYNINQPLTNLKYLNSERLCFSNAVMGKI...

Figure 4: The data obtained from Swiss-Prot, the supplementary file obtained from [8]

The paper also had a ready to use supplementary file of pre-trained k-mer embeddings, where overall 100 dimensional vectors for 9047 3-mers were listed. The ProtVec is only an embedding method that can later be used for various tasks as protein visualization, structure prediction, domain extraction, and interaction prediction. The specific task that was explored in this capstone to test the quality of embeddings obtained is the protein family classification task. The original paper used a support vector machine classifier (SVM) to perform protein family classification using only sequence (primary structure). The authors evaluated K-nearest neighbors in 2xfold cross-validation and obtained the optimal results as following: window size of 3, embedding vector size of 100, and a context size of 25, which is not common for tasks in NLP. They used all the Swiss-Prot database available at the time of publication of 546,790 sequences and using window size 3 the corpus was 1,640,370 sequences of 3-grams. Then this corpus was trained using the skip-gram methods together with the negative sampling technique that was discussed in the previous section.

Family information was obtained using the Protein Family Database (Pfam). To train the skip-gram an equal number of positive and negative examples were chosen for each family type and then 10xfold cross-validation was applied to SVM and metrics were obtained from the $confusion\_matrix$ available at $sklearn.metrics$ in python.

$$\text{Specificity/Precision} = \text{TP rate} = \frac{TN}{TN+FP}$$
$$\text{Sensitivity} = 1 - \text{FP rate} = \frac{TP}{TP+FN}$$
$$\text{Accuracy} = \frac{TN+TP}{P+N}$$

Here TP and FP rates stand for true positive and false positive, respectively. A true positive is an outcome where the model correctly predicted the protein family label for positive examples and false positive is an outcome where the model incorrectly predicts the positive class.

3.1 Data

Overall in the supplementary files provided to the original paper, there were in total 324,018 sequences. Let's explore the provided data in more detail.

The most frequent protein family is 50S ribosome-binding GTPase with 3084 sequences. However, the overall distribution is far from being unifrom, the top 100 most frequent family types from overall 7027 constitute 95514 out of 324018, which is approximately 1/3 of all sequences. The 980 family types have only representative sequence present in this dataset.

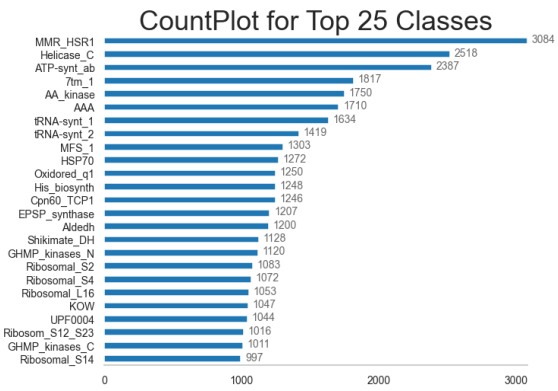


Figure 5: The bar graph for the top frequent 25 protein families, the subscript is FamilyID

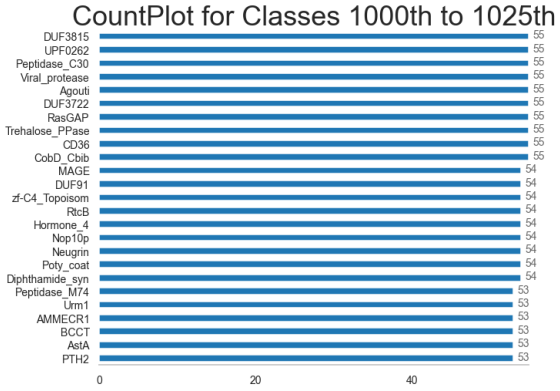


Figure 6: The bar graph for the top frequent 1000-1025th protein families, the subscript is FamilyID

As you may see from Figure 6, the number of sequences for the 1000th top frequent family is only 55 sequence examples. The number of example sequences in a protein family dropped significantly compared to the top 25 protein families, where protein families had sequences ranged from 997 to 3084.

The length of the sequences in the dataset also vary a lot, the maximum length of single sequence is 22,152 which comes from SEA protein family and the minimum is only 7 amino acids: 'WREMSVW' from WWamide peptide family. The distribution on the figure has tails, but since the overall number scales up, they become insignificantly small compared to major trend.

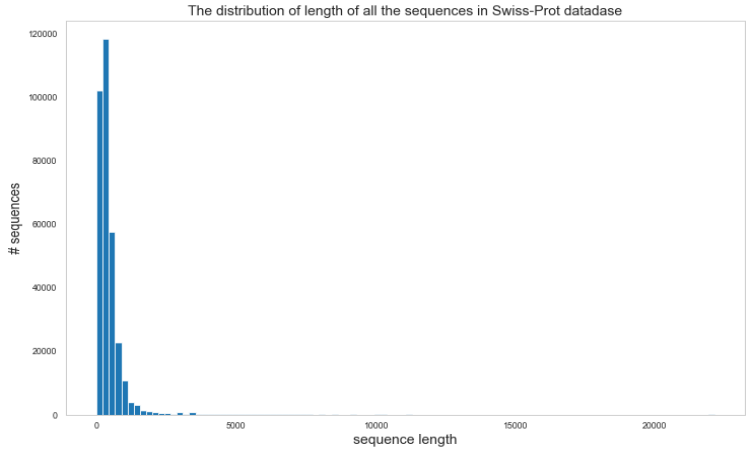


Figure 7: Histogram for distribution of length of protein sequences for entire dataset

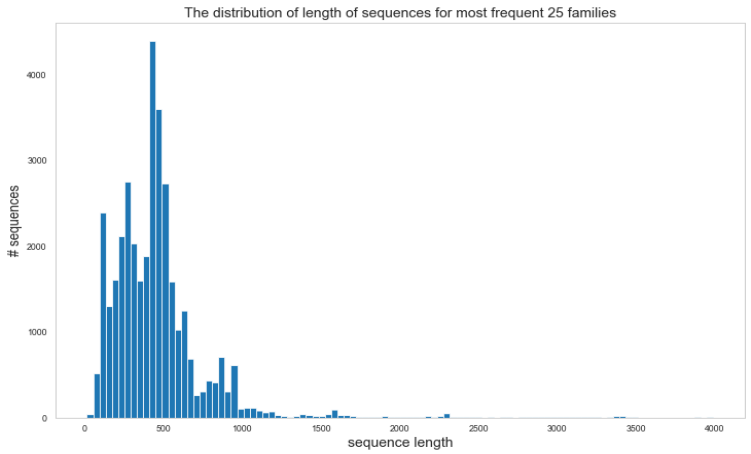


Figure 8: Histogram for distribution of length of protein sequences for top 25 protein families

3.2 Experiments

For the following experiments, the implementation of Word2Vec in the gensim library was primarily used. The data that I use to compare the results for is the same that was used in the original Asgari et al., and they were taken from the supplementary files attached to the paper.

The analysis for the quality of embeddings was accessed using the accuracy, sensitivity, and precision scores. The original paper also analyzed the protein-space to further understanding of the biochemical and biophysical implications. However, the same was not realized in this capstone project due to insufficient competency in biology and protein structure.

Using the module of python named biovec, which is python implementation for ProtVec algorithm based on gensim I obtained the vector representation for the top 10 and top 25 most frequent families. The settings which were used are identical to original paper, and reducing the dimensions of the obtained vectors to 2, we can clearly see the main hypothesis of the NLP, that words(each represented as n-dimensional vector) with similar meaning should be close to each other in n-dimensional vector space. From figure 9, as can be seen in the upper right corner, the vectors from same class are grouped together, even though there are definitely some overlaps. The graphs were obtained through application of dimensionality reduction techniques. The dimensions were reduced from 100 dimensional vectors to 2D through PCA (Principal component analysis) and t-SNE (t-Distributed Stochastic Neighbor Embedding) and visualized.

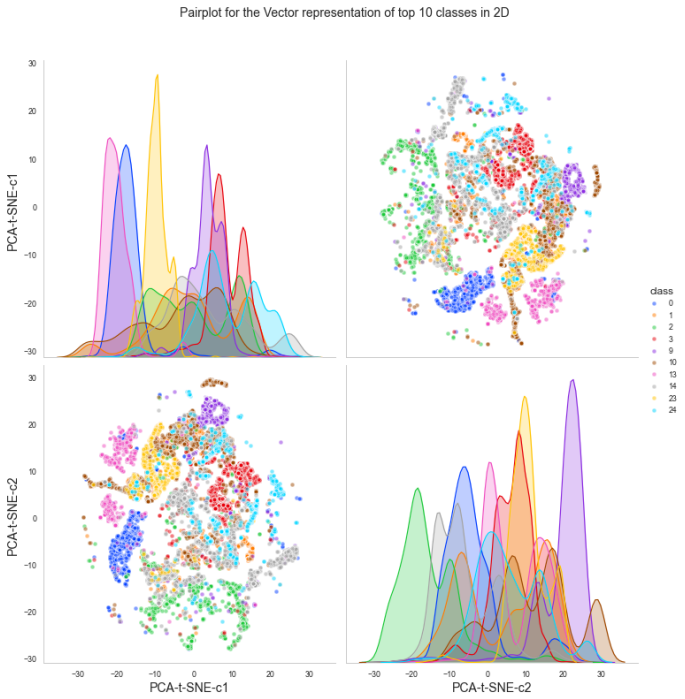


Figure 9: Pairplot for the vector representation in 2d for the sequences of 10 most frequent families

But if consider less frequent sequences like Ribosomal S14 - 997 sequences, Ribosomal S11 - 980, Ribosomal L13 - 705 and compare between 3 classes only, making the same plotting one can clearly see the separation between them on figure 10.

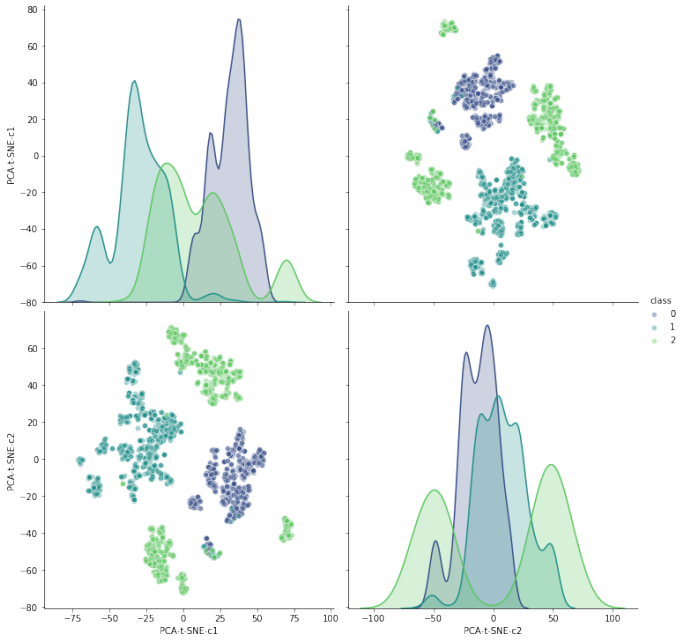


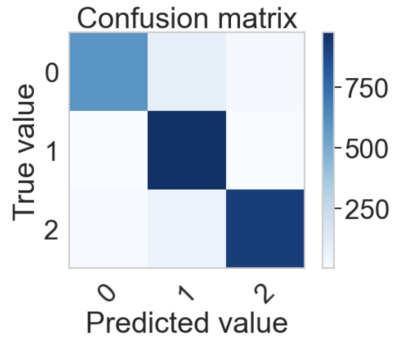
Figure 10: The pairplot for only 3 family types, the sequences are clearly grouped

Using these 3 classes doing multi-class classification resulted in 93% accuracy:

```
print("Prediction accuracy: %.2f%%" % (accuracy_score(df_subset['class'], probs)*100))
```

```
Prediction accuracy: 92.99%
```

```
confusion_matrix_plot(cm, names=[0,1,2]);
```




```
print(classification_report(df_subset['class'], prob_class,labels=[0, 1, 2]))
```

	precision	recall	f1-score	support
0	0.98	0.85	0.91	705
1	0.87	0.99	0.92	980
2	0.97	0.93	0.95	997
accuracy			0.93	2682
macro avg	0.94	0.92	0.93	2682
weighted avg	0.94	0.93	0.93	2682

Figure 11: Confusion matrix and metrics for 3-class classification

4 Existing modifications of ProtVec for protein family classification task

In general, the protein classification is done using methods from two big groups: alignment and alignment-free. They fundamentally differ in their approaches to the classification task. However, the details of the alignment method's work will not be discussed in this capstone, and they can be found in [10]–[12].

The most popular alignment methods:

- BLAST
- CLUSTALW
- Profile Hidden Markov Model

Alignment-free methods are 'approaches that avoid the costly reliance on dynamic programming in favor of faster alternatives, usually based on the tokenization of the sequence into a set of substrings or k-mers, words of length k extracted from the sequence' [12]. To overcome the main disadvantage of computational complexity the new approaches, which are generally all grouped in alignment-free methods appeared. As the computational power grew and powerful computers became more widely spread than ever, the usage of more complex models based on deep learning has been trending. This project primarily focuses on the methods that use embeddings for the classification task. The general tactic of embedding the biological sequences is as follows.

- First, obtaining a corpus. Usually, done through a k-mer representation of overlapping sets (k=3-6).
- The Second step is to find an alternative representation in a new domain (like R^n) where a formal distance metric is defined such that the pairwise similarity is preserved under new representation.

4.1 ProtDocVec/Seq2Vec

4.1.1 Doc2Vec

As follow-up work, Mikolov T. and Le Q. (2014) proposed a generalized version of Word2Vec named doc2vec which produces "fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents" [13] Now, in addition to the word vectors, there is a "Paragraph Vector", unique for each paragraph/document/etc. The word vectors represent the meaning of words and paragraph vector acts as a topic of a document. It also includes two models analogously to Skip-Gram and CBOW we have, Distributed Memory Model of Paragraph Vectors (PV-DM) and Distributed Bag of Words version of Paragraph Vector (PV-DBOW), respectively.

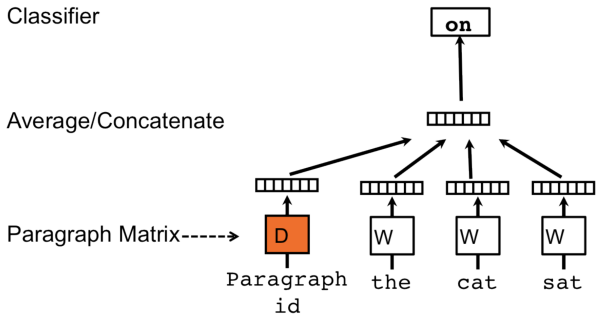


Figure 12: PV-DM
Source: Mikolov and Le, 2014 [13]

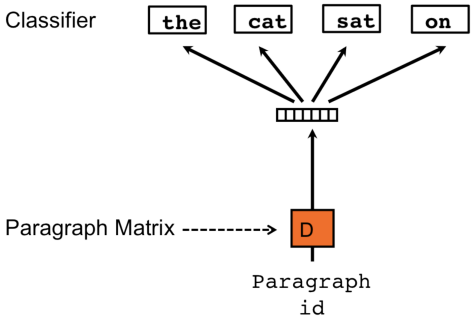


Figure 13: PV-DBOW
Source: Mikolov and Le, 2014 [13]

The primary advantage of doc2vec model is that unlike the Word2Vec, it can capture meaning from the word ordering, i.e. the vector output for biological sequence will be different if given in other order. In addition, authors note during experiments they showed that PV-DM (analogy of Skip-gram) is consistently better than PV-DBOW [13]. It is important to note that, when using doc2vec, the word vectors are global thus they update for each context, while Paragraph ID or document vectors are local and only updated for contexts from this document.

4.1.2 Seq2Vec

Using a similar idea to create embeddings the two methods were created independently, however now instead of using the Word2Vec model the doc2vec (see the previous section) model is used. The two methods - ProtDocVec and Seq2Vec - were crated independently, by Nambiar A. [14] and Kimothi D. [9] and his co-authors. Both methods exploit the same technique, compared to ProtVec which bases on the Word2Vec, the Seq2Vec (or ProtDocVec) are based on the generalized version of Word2Vec named doc2vec, which generates embedding for the whole document, in this specific case, it created one embedding for the whole sequence directly.

The authors tried to enhance ProtVec by eliminating one potential weakness: the protein embeddings don't fully capture the order of the amino acids in a sequence. Since the

protein sequence is a combination of a restricted alphabet of 20 amino acids the order of occurrence determines important biophysical and biochemical properties. Experimenting with the ProtVec as can be seen in the figure below, the identical sets of amino acids outputs very high cosine similarity. Despite how the amino-acids were shuffled differently from the original sequence seen on the first row, the cosine similarity was in the range of 0.98 - 0.99. Thus, the change of training model to the doc2vec now should theoretically solve this issue.

The pre-processing of the protein sequence was done in the same way as in the original paper.

```
a=sum(pv2.to_vecs("ATATQSQSMTEELIPDFTPALQ"))
```

```
b=sum(pv2.to_vecs("TDFATEELITAPALPQTQSQSM"))
```

```
from numpy import dot
from numpy.linalg import norm

cos_sim = dot(a, b)/(norm(a)*norm(b))
cos_sim
```

```
0.98233557
```

Figure 14: Identical set of amino acids will output very similar protein embedding, even though the biological properties of two may vary

In an attempt to solve this issue, Kimothi et al. tried to apply doc2vec (see the previous subsection) model instead of Word2Vec and named a new model as seq2vec. Applying, PV-DM analogously as Skip-Gram they used the kNN (k-nearest neighbor algorithm) to identify the hyperparameters and used a multi-class SVM (Support Vector Machine) classifier with a linear kernel (used C=1 for Seq2Vec and C=7.5 for ProtVec) and applied the one-vs-rest strategy.

The exact choice of SVM hyper-parameters, including the type of kernel and C, are not mentioned in [2]. We chose to use an SVM classifier with linear kernel and C equal to 1.0, these choices being made based on grid search. We did three sets of experiments with 1000, 2000, and 3000 biggest families. The results reported in Table 1 show that the accuracy and specificity obtained using Seq2Vec is consistently better than ProtVec while ProtVec performs slightly better in terms of sensitivity than Seq2Vec.

Note that this experimental setting provides a much harder challenge than the earlier binary classification and the results prove that the embeddings learned using Seq2Vec are superior t learned using ProtVecs for this task.

To assign a family to a test sequence, we find its k-nearest training sequences in the vector space and predict the family using a majority vote. Here we assume that we embed both the train and test data simultaneously. If test data is not available for embedding, we can later use gradient-descent to learn the vector representation of the test sequences as explained in [14].

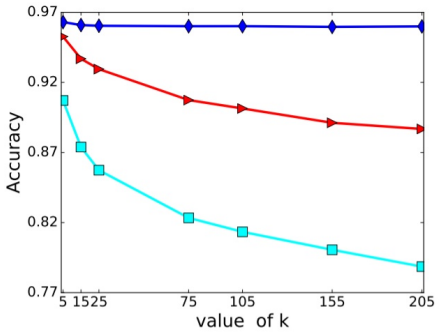
	# of classes	Specificity (%)	Sensitivity (%)	Accuracy (%)
seq2vec	1000	**97.49 ± 0.05**	92.72 ± 0.06	**95.10 ± 0.05**
	2000	**97.01 ± 0.04**	93.07 ± 0.07	**95.04 ± 0.05**
	3000	**96.67 ± 0.05**	93.18 ± 0.08	**94.92 ± 0.06**
ProtVec	1000	91.61 ± 0.05	**95.40 ± 0.04**	93.50 ± 0.05
	2000	90.10 ± 0.07	**95.75 ± 0.05**	92.92 ± 0.05
	3000	88.61 ± 0.09	**95.95 ± 0.05**	92.27 ± 0.06

Figure 15: Performance in Binary Classification

	Precision(%)	Sensitivity (%)	Accuracy (%)
seq2vec	**83.37 ± 0.052**	**81.69 ± 0.067**	**81.29 ± 0.057**
ProtVec	79.01 ± 0.071	76.78 ± 0.082	76.70 ± 0.080

Figure 16: Performance of Seq2Vec vs. ProtVec in Multiclass Classification
Source: Kimothi et al., 2016 [9]

The authors then compared the performance of Seq2Vec and ProtVec to an alignment method BLAST and obtained the following results.



kNN Classification: Performance of seq2vec (–▶–▶–) and ProtVecs (–■–■–) as compared to that of BLAST (–♦–♦–) as a function of k used for kNN.

Figure 17: Accuracy compared with alignment method BLAST
Source: Kimothi et al., 2016 [9]

The overall tendency of stability of alignment methods such as BLAST (Basic Local Alignment Search Tool), pHMM(Profile Hidden Markov Model) is appealing to researchers since it ensures the accuracy which does not fluctuate over hyperparameters or sequence length. However, the stability comes with the price, in this particular case it is a retrieval or classification time that encouraged the development of alignment-free methods, and in particular, embedding based methods such as ProtVec and others. In the next section, the comparison table of time for different classification methods will be shown, together with a more advanced method named DeepFam.

4.2 ProtVecX

ProtVecX - an extended version of ProtVec proposed by original authors, which also uses shallow skip-gram neural network but now with fasttext [15], [16] rather than Word2Vec, and they do not simply create overlapping k-mers, but apply a technique from byte-pair encoding (used for data compression-replaces the most frequent byte with unseen data) and introduce peptide-pair encoding (PPE) algorithm [17]. The vectors are 500 dimensional. However, compared with the ProtVec it does not outperform it according to precision and recall score, and has only slightly better macro F1-score(increase by 0.01). Therefore due to slight improvement, this method was not considered in this capstone project for comparison purposes.

4.3 DeepFam

The DeepFam was proposed by Seo S. as part of a graduate thesis at Seoul National University [18]. This method is a deep learning-based alignment-free method used not only for protein family prediction but also for modeling. The architecture of DeepFam is shown below:

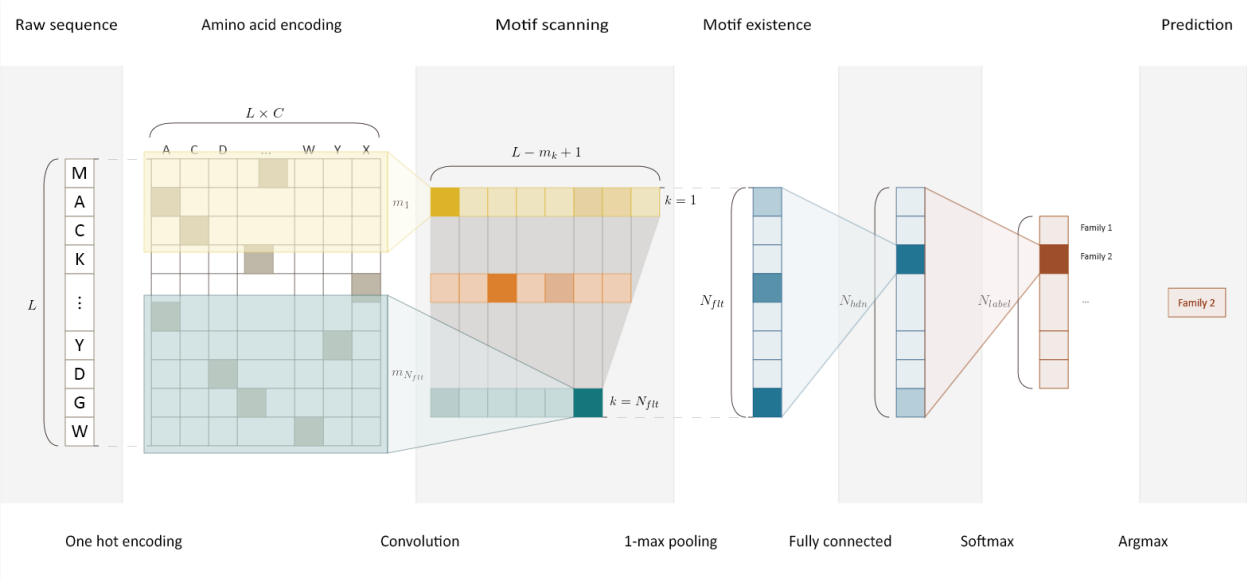


Figure 18: DeepFam architecture
Source: Seo et al., 2018 [18]

The overall performance for absolute fairness was based on databases on which the 4 mentioned methods were not tested before. The Clusters of Orthologous Groups (COGs) and G Protein-Coupled Receptor (GPCR) were used for evaluation. Each COG family has a different number of proteins which also vary in length, after pre-processing which included the elimination of low frequent data that had less than 100, 250, and 500 sequences. Therefore we have corresponding datasets COG-500, COG-250, COG-100. The GCPR dataset included 5 families and 8,222 protein sequences. Here the classifier used was not SVM as in original paper but logistic regression.

Dataset	COG-500-1074	COG-250-1796	COG-100-2892
DeepFam	**95.40**	**94.08**	91.40
pHMM	91.75	91.78	**91.67**
3-mer LR	85.59	81.15	75.44
Protvec LR	47.34	41.76	37.05

Figure 19: Performance of DeepFam compared to ProtVec
Source: Seo et al., 2018 [18]

The time elapsed on training are shown below. This clearly shows the advantage of alignment-free methods.

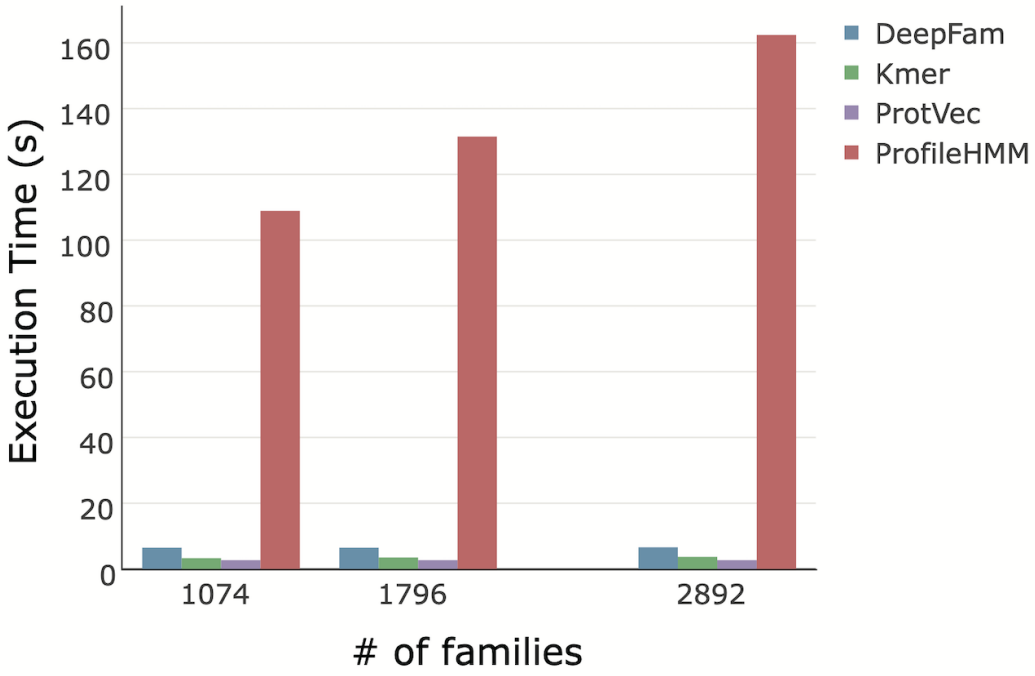


Figure 20: The average elapsed time of five trials to predict families of 1,000 protein sequences for each method.
Source: Seo et al., 2018 [18]

As can be seen in the figure above the ProtVec was the fastest among the 4 models, while pHMM (profile Hidden Markov Model) took 20-40 times longer than ProtVec. Low-level features are now proven to be able to classify proteins with reasonable accuracy. Different hypothesis like the proteins which have common ancestor is more easily classified, while ones that do not have common roots but only functions are more complex have yet to be checked using deep learning methods.

5 Conclusion

The protein family classification task has been one of the most encouraging tasks after the revolution of Next generation sequencing (NGS), which increased the availability of data such as the primary structure of proteins. The computationally expensive alignment methods which existed before the advancement of neural networks have dominated this field. But now, the alignment-free methods which are developing and using the advantage of neural networks have raised the baseline in this field. The comparatively simple embedding methods like ProtVec are considered to be fast and easy to use for a large amount of corpus. However, due to the limitations of Word2Vec and the basic k-mer model together with heavy reliance on hyperparameters, adequate accuracy is not satisfactory anymore. Therefore, several modified versions based on the initial idea of ProtVec are now employing advanced and more complex methods to increase stability and accuracy. The Seq2Vec which has an average accuracy of 81.29% for multiclass classification of top 25 families in the Swiss-Prot database outperformed the ProtVec which obtained only 76.70%. The further improvement of Seq2Vec using different metrics besides Euclidean, namely Mahalanobis metric also improved the performance to 92.44% when having a dimension size=250 and to 87.42% when having dimension size=100, while ProtVec with same metric and dimension size=100 obtained 87.12%.
The concluding points:

- The embedding methods - low-level features - are powerful enough to classify proteins with reasonable accuracy
- Embedding methods use only the primary structure of protein - sequence, there is no need for secondary or tertiary structure
- The training of ProtVec embedding models is computationally inexpensive and fast, compared to alignment-based methods - it took 2h 36 min to train on whole Swiss-Prot and requires one training only.
- The results are interpretable and can be visualized by using the dimensionality reduction methods like t-SNE and PCA.
- Obtained embeddings can be used for pre-processing of large scale databases.
- Further improvements can be made using a different metric, or hybrid approaches.

The code implementations used in this project are available at github.com/damilya99/Capstone.

5.1 Future Work

The Seq2Vec has been later improved using different metrics other than Euclidean, namely Mahalanobis distance by Kimothi et al. [19] and has shown the improved performance of embeddings in the classification task. Also, a recent paper that has been published during the work on this capstone, in March 2020 proposed a novel method named SuperVec which is an advanced version of Seq2Vec [20]. SuperVec is a supervised learning method that uses

fasttext and employs both contextual information as well as the class-label information to learn sequence embeddings. Further work will be to explore different metrics and techniques to improve the ProtVec and Seq2Vec embeddings.

References

[1] *Why classify proteins?* June 2016. URL: https://www.ebi.ac.uk/training/online/course/introduction-protein-classification-ebi/protein-classification/why-classify-proteins.

[2] Christine A Orengo and Janet M Thornton. "Protein families and their evolution-a structural perspective". In: *Annual review of biochemistry* 74 (2005), pp. 867–900. ISSN: 0066-4154. DOI: 10.1146/annurev.biochem.74.082803.133029. URL: https://doi.org/10.1146/annurev.biochem.74.082803.133029.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 1137–1155. ISSN: 1532-4435.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in Neural Information Processing Systems* (2013), pp. 1–9. ISSN: 10495258. arXiv: 1310.4546.

[5] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: Jan. 2008, pp. 160–167. DOI: 10.1145/1390156.1390177.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, et al. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781 (2013). URL: http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781.

[7] McCormick C. *Word2Vec Tutorial Part 2 - Negative Sampling*. Jan. 2017. URL: http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/.

[8] Ehsaneddin Asgari and Mohammad RK Mofrad. "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics". In: *PloS one* 10.11 (2015), e0141287.

[9] Dhananjay Kimothi, Akshay Soni, Pravesh Biyani, et al. "Distributed Representations for Biological Sequence Analysis". In: *CoRR* abs/1608.05949 (2016). arXiv: 1608.05949. URL: http://arxiv.org/abs/1608.05949.

[10] Marc A Marti-Renom, MS Madhusudhan, and Andrej Sali. "Alignment of protein sequences by their profiles". In: *Protein science : a publication of the Protein Society* 13.4 (Apr. 2004), pp. 1071–1087. ISSN: 0961-8368. DOI: 10.1110/ps.03379804. URL: https://europepmc.org/articles/PMC2280052.

[11] Michael Heinzinger, Ahmed Elnaggar, Yu Wang, et al. "Modeling aspects of the language of life through transfer-learning protein sequences". In: *BMC Bioinformatics* 20 (2019).

[12] Zhiqiang Zeng, Hua Shi, Yun Wu, et al. "Survey of Natural Language Processing Techniques in Bioinformatics". In: *Computational and Mathematical Methods in Medicine* 2015 (2015). ISSN: 17486718. DOI: 10.1155/2015/674296.

[13] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053 (2014). arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

[14] Ananthan Nambiar. *Computing the Language of Life*. 2019.

[15] Piotr Bojanowski, Edouard Grave, Armand Joulin, et al. "Enriching Word Vectors with Subword Information". In: *CoRR* abs/1607.04606 (2016). arXiv: 1607.04606. URL: http://arxiv.org/abs/1607.04606.

[16] Armand Joulin, Edouard Grave, Piotr Bojanowski, et al. "Bag of Tricks for Efficient Text Classification". In: *CoRR* abs/1607.01759 (2016). arXiv: 1607.01759. URL: http://arxiv.org/abs/1607.01759.

[17] Ehsaneddin Asgari, Alice McHardy, and Mohammad R.K. Mofrad. "Probabilistic variable-length segmentation of protein sequences for discriminative motif mining (DiMotif) and sequence embedding (ProtVecX)". In: *bioRxiv* (2018). DOI: 10.1101/345843. eprint: https://www.biorxiv.org/content/early/2018/07/12/345843.full.pdf. URL: https://www.biorxiv.org/content/early/2018/07/12/345843.

[18] Seokjun Seo, Minsik Oh, Youngjune Park, et al. "DeepFam: Deep learning based alignment-free method for protein family modeling and prediction". In: *Bioinformatics* 34.13 (2018), pp. i254–i262. ISSN: 14602059. DOI: 10.1093/bioinformatics/bty275.

[19] Dhananjay Kimothi, Ankita Shukla, Pravesh Biyani, et al. "Metric learning on biological sequence embeddings". In: *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)* (2017), pp. 1–5.

[20] Dhananjay Kimothi, Pravesh Biyani, James M Hogan, et al. "Learning supervised embeddings for large scale sequence comparisons". In: *bioRxiv* (2019). DOI: 10.1101/620153. URL: https://www.biorxiv.org/content/early/2019/04/26/620153.