Analysis of NLP Techniques for Protein Family Identification

Beethika Tripathi[1], Karthik Azhagesan[2] , Mohnish Uyyuru[3]

[1]CS15S004, [2]BT15S001, [3]CS13B019

Abstract. We aim to demonstrate that continuous space representations can perform better than traditional models that rely on discrete features. A drawback of traditional statistical models is that the use of discrete features causes the data to become sparse due to the curse of dimensionality. In continuous space representations however this problem is resolved. The continuous space representations of protein sequences have a wide array of applications in bioinformatics such as family classification, protein visualization, structure prediction, disordered protein identification, and protein-protein interaction prediction. However, for our project we have chosen to compare the performance only on the protein classification task.

1 Introduction

Understanding the structure, dynamics and function of proteins strongly parallels the mapping of words to meaning in natural language. The words in a text document map to a meaning and convey rich information pertaining to the topic of the document. Similarly, protein sequences also represent the "raw text" which are motifs and carry high-level information about the structures, dynamics and functions of proteins. For example, the document topic "sports" can be found by words such as ball and ground. Similarly, we can find the protein family "Tropomyosin" by motifs that can be obtained from the primary sequence of all the sequences in that protein family.

Like strings of letters, protein sequences are linear chains of amino acids. Amino acids are fundamental building blocks in protein sequences. They can be one of 20 different types, corresponding to different chemical structures labeled as (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y). There are hundreds of different scales of properties of amino acids, including size, hydrophobicity, electronic properties, aromaticity, polarity, flexibility, secondary structure propensity and charge to name just a few. Thus, although the 20 amino acids are a reasonable starting point to define building blocks in protein sequences, smaller, larger or uniquely encoded units may often be functionally more meaningful. Most functions in biological systems are carried out by proteins mainly transmission of information, for example in signaling pathways, enzymatic catalysis and transport of molecules. Proteins are synthesized from small building blocks, amino acids, of which there are 20 different types. The amino acids are connected to form a string (Primary Structure). The three-dimensional structure has certain

repetitive elements known as secondary structures. α-helices, β-strands are highly occurring secondary structures and are connected by different types of loops.

Many of the hallmarks of statistical analysis of biological sequences are similar to those of human languages. Large data bodies need to be analyzed in both cases. Fundamental units of human languages include higher order structures, paralleled by domains, subunits or functionally linked proteins. Computer-based derivation of meaning from text is analogous to the prediction of structure and function from primary sequence data. This has been the motivation for our application of NLP techniques to the protein family classification problem.

2 Methods

2.1 Naïve Bayes Classifier

Naïve Bayes classifier (Bai et al. 2004) uses all the words in the text as features. It assumes given the class the feature vectors are independent of each other. It assumes a set of predefined classes {… c_i, …}, and given a text w = w_1,...,w_2 , it computes the posterior probability that the text belongs to all classes in the set and then assigns the text to the class(es) with the highest probability value(s). The posterior probability is computed using the Bayes rule:

$$P(c_i|w) = \frac{P(w|c_i)P(c_i)}{P(w)}$$

here the P(w) can be considered as a normalizing factor which is independent of classes so can be ignored for ranking the class for a text.

Using Naïve Bayes assumption, the likelihood term can be re-written as follows:

$$P(w|c_i) = \prod_{j=1}^{m} P(w_j|c_i)$$

Therefore, the posterior can be re-written as follows:

$$P(c_i|w) \propto \prod_{j=1}^{m} P(w_j|c_i)P(c_i)$$

The prior in equation (1) can estimated by the percentage of training examples that belong to class c_i .

Laplacian Smoothening is used to compute the likelihood, to avoid the zero-likelihood problem. This is given as:

$$P(w_j|c_i) = \frac{1 + count(w_j, c_i)}{|V| + N_i}$$

here count(w_j,c_i) is the number of times word w_j occurs in class c_i, |V| is the number of unique words .

2.2 Statistical Language Model

Statistical Language Model or Ngram Model (Jurafsky and Martin 2014) approximates the probability of the n^{th} word on the basis of previous n-1 words. This assumption comes from the markovian assumption that the probability of some future unit ca be predicted without looking too far into the past. So the probability of text say w composed of words w_1 w_2 w_3 … w_n can be written as:

$$P(w) = P(w_1\ w_2\ w_3\ ...\ w_n) = \prod_i P(\ w_i|w_1\ w_2\ w_3\ ...\ w_{i-1})$$

The language model is constructed for each class. Then the class for which it gets the highest score, the text is classified into that class.

Naïve Bayes model is also a sort of language model just the score computation is bit different.

2.3 Katz Backoff Model

Katz's backoff model (Jurafsky and Martin 2014) is a generative n-gram language model with a bit modification. This model is superior to a regular n-gram model because it recursively "backs off" to an (n-1)-gram model when the evidence of the given n-gram is less than a particular threshold. Thus, for any instance, the model with the most reliable evidence is used for prediction. Usually the threshold used is 0, i.e., n-grams that have never occurred in training data. The model outputs the probability of a word occurring in a sentence given the (n - 1) words occurring before the position under consideration.

2.4 Word2Vec

Word2Vec (Mikolov et al. 2013) is an architecture that use neural networks to learn the continuous vector representations for words. Each word is mapped to a vector in embedded space and these vectors are updated such as to minimize the distance of the word’s vector to the vector of the words with which it frequently co-occurs. It comes in two variants, the CBOW and the Skip-gram models.

Continuous Bag of Words

The model consists of three layers: input, hidden and output. Assuming a window size of N, the neural network has N X V sets of nodes in its input layer, where V is the size of the vocabulary.

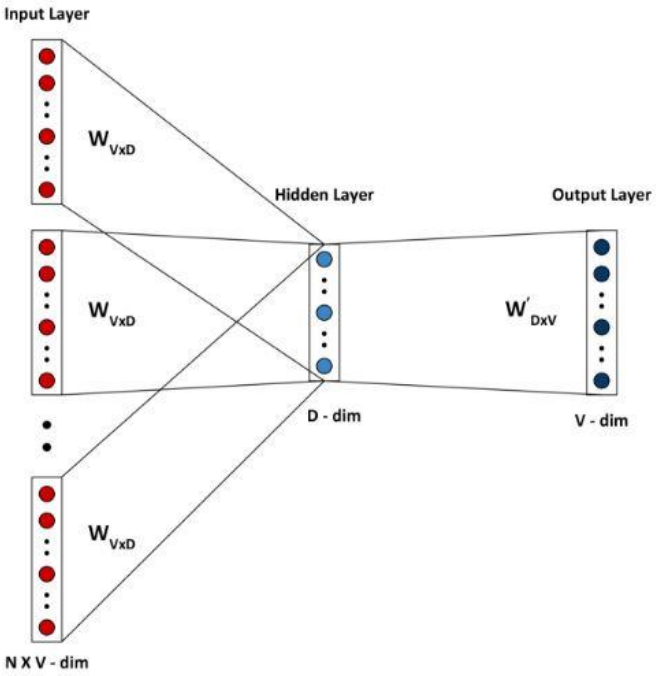

Fig. 1. Architecture of CBOW model

The hidden layer has D nodes, where D is the dimension of the output vector embeddings. The output layer has V nodes. The input layer is basically made up of N sets of V nodes each. The weight matrix between each set of input nodes and the hidden layer is same. Each node in the output layer stands for a word in the vocabulary. Each set of nodes belongs to one word of the context and is fully connected to the hidden layer. The hidden layer is fully connected to the output layer. Let the input to hidden weights matrix be W and the hidden to output weights matrix be W'. W is a V X N matrix and W' is an N X V matrix.

The goal is to learn the latent representation of words such that given the context words i.e. the input layer it can predict the target word i.e. output layer. The objective function can be written as follows:

$$minimize\ -log\ P(w_t|\{\Phi(w_{t-c}),\dots,\Phi(w_{t-1}),\Phi(w_{t+1}),\dots,\Phi(w_{t+c})\}$$

where c is the context window and Φ is the representation learnt for the word.

Skip-gram

This is just the reverse of CBOW model, where given the word it should predict the context word,

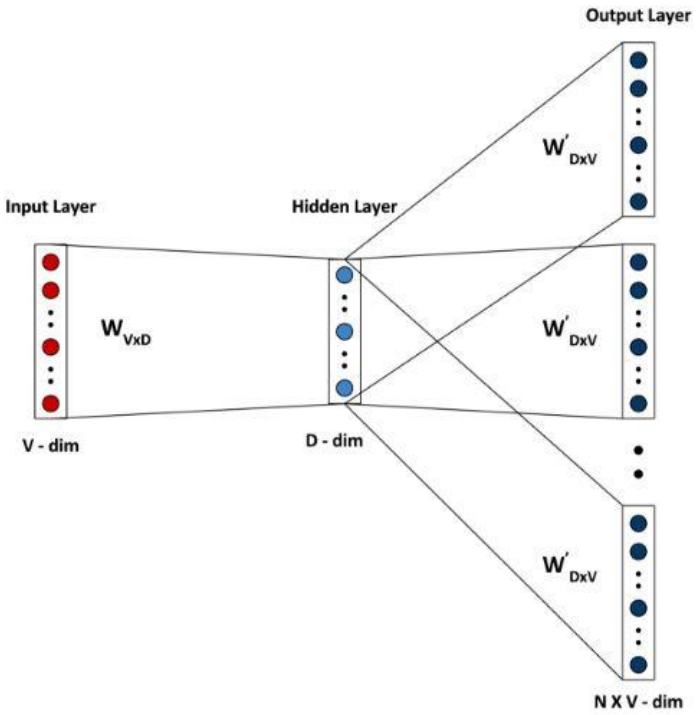

Fig. 2. Architecture of Skip-gram model

The objective function can be given as follows:

$$minimize\ -log\ P(\{w_{t-c}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+c}\}|\Phi(w_t)$$

The conditional likelihood can be computed using hierarchical softmax or negative sampling which can be found in (Mikolov et al. 2013).

2.5 **ProtVec** (Asgari and Mofrad 2015)

An approach similar to what is used in word2vec is used to create continuous space representations of protein vectors. The first step in training the protein vectors is to break the sequences into subsequences (i.e. biological words). The technique used in this representation is that of splitting the protein sequence into 3 non overlapping tri-gram sets. This so-called embedding model needs to be trained only once and may then be adopted in feature extraction part of specific problems.
We generate 3 lists of shifted non-overlapping words, as shown below:

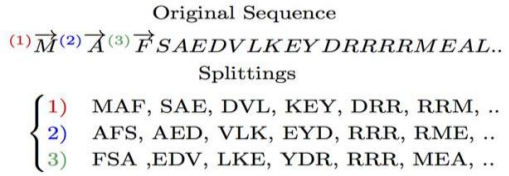

Fig. 3. Overlapping Tri-grams

These sequences become the sentences and word2vec is used to learn the representation of each trigram. Once it is done then these vectors for each trigram are added to get the representation of the sequence. Which can be used for classification of families.

2.6 Segmentation of sequences using Viterbi Algorithm

Minimum Description Length (Robinet 2004) is a formalization of Occam's razor principal which states that the best compression of the data is achieved by the best hypothesis.

Viterbi algorithm (Kit and Wilks 1999) using description length gain finds the shortest path through graphs. In the graph, each edge is a word and edge weight is negative log probability. The shortest path is the one with the best segmentation.

In Viterbi algorithm there are two steps:

1. Forward step: In this step, we find score of best path to each node.
2. Backward step: In this step, we create the best path.

3 Experiments

We have taken 546,790 protein sequences from Swiss-Prot database. About 324,018 sequences have protein family information. There are about 7,027 protein families. For learning representations we consider the complete corpus. But for classification we consider only the ones which have family information. We are doing one-vs-all classification. The positive sequences are the sequences from the family we are considering and the same amount of negative sequences are randomly sampled from other families. As the number of families are too large so we have considered a subset of families which have varied number of samples. As one above 3,000 samples one with samples in the range of 2,500 – 2,000 and so on.

For the representation learning methods we have used Support Vector Machines (SVM) with Gaussian kernel for classification and have used grid search to find out the best set of parameters for classification.

For evaluation we have done 10-fold cross validation and reported specificity, which is the true negative rate, sensitivity, which is the true positive rate, and accuracy of family.

$$Sensitivity = True\ Positive\ Rate = \frac{TP}{TP + FN}$$

$$Specificity = True\ Negative\ Rate = \frac{TN}{TN + FP}$$

$$Accuracy = \frac{TP\ +\ TN}{P + N}$$

where TP means true positive, TN means true negative, FP means false positive, FN means False Negative, P means positive and N means negative.

3.1 Naive Bayes Classifier

Naïve Bayes classifier with multinomial likelihood is used to compute the posterior of sequence given positive and negative family, one which has the highest score the sequence is classified to that family.

3.2 Katz Back-off Language Model

The Katz back-off language model is used to get the average log likelihood of all the trigrams in the corpus. This is computed for both the positive and negative model. Then the test sequence is assigned a class to the model that got a lower score.

3.3 Word2Vec using overlapping trigrams:

We created overlapping trigrams of the entire protein sequence as follows:

Example:
If the sequence is as follows:

MDDKFTTLPCELEDYPGSITCPHCSAQITTAVDHVVGKMSWVVCTAITLACL
PCCCIPFLCNSTKDVRHTCPKCKQAVFVYKIL

then the overlapping trigrams are as follows:

MDD	KFT	TLP	CEL	EDY	PGS	ITC	PHC	SAQ	ITT
	AVD	HVV	GKM	SWV	VCT	AIT	LAC	LPC	CCI
	PFL	CNS	TKD	VRH	TCP	KCK	QAV	FVY	KIL
DDK	FTT	LPC	ELE	DYP	GSI	TCP	HCS	AQI	TTA
	VDH	VVG	KMS	WVV	CTA	ITL	ACL	PCC	CIP
	FLC	NST	KDV	RHT	CPK	CKQ	AVF	VYK	

DKF	TTL	PCE	LED	YPG	SIT	CPH	CSA	QIT	TAV
	DHV	VGK	MSW	VVC	TAI	TLA	CLP	CCC	IPF
	LCN	STK	DVR	HTC	PKC	KQA	VFV	YKI	

Then we ran word2vec (implementation of gensim models) on the corpus of 546790 sequences for a context window of 25 and for 100 dimensions. This gave us all trigram representation of all 9424 trigrams. Then we summed up the representation of all individual overlapping trigrams in the sequence (3 different trigrams for each sequence) to get a trigram representation for the entire sequence. This gave us a summed up representation for all sequences for which families were present. Now this representation can be used for classification.

3.4 Word2Vec using segments:

We created frequency of Ngrams upto 10 grams in the preprocessing step. This enabled us to create segments of length upto 10 faster since the original Viterbi algorithm complexity was of O(mn) and when we did this preprocessing it enabled us to segment faster for 546790 sequences. Also we restricted the segment size to 10 since in biology motifs were found to be of size 3-10 residues (Neduva and Russell 2006).
If the sequence is as follows:

MDDKFTTLPCELEDYPGSITCPHCSAQITTAVDHVVGKMSWVVCTAITLACL
PCCCIPFLCNSTKDVRHTCPKCKQAVFVYKIL

then the segments are as follows:

M	D	D	K	F	T	T	L	P	C
	EL	E	D	Y	P	G	S	I	
	TCPHC	S	AQ	I	T	T	A	V	D
	H	V	V	GK	M	S	W	V	V
	C	T	A	I	TL	A	C	L	PCCC
	I	P	F	L	C	NS	T	K	D
	V	R	H	TCPKCK		Q	A	VF	V
	Y	K	I	L					

Here we are ignoring the unigram while learning the word2vec representation. Learning unigram and bigram will add a lot of noise to the model as they will capture a very generalized representation as they frequently occur so their neighborhood is varied.

Once we have the representation method explained in section 3.3 can be used for classification.

3.5 Word2Vec using overlapping trigrams and segments:

Once the segmentation is done then again the overlapping trigram approach is used to find out 3 sequences from a single sequence. The segments are considered as it is in all the 3 sequences and the overlapping trigrams are found out on the remaining unigrams.

If the sequence is as follows:

MDDKFTTLPCELEDYPGSITCPHCSAQITTAVDHVVGKMSWVVCTAITLACL
PCCCIPFLCNSTKDVRHTCPKCKQAVFVYKIL

then the overlapping trigrams with segments are as follows:

MDD	KFT	TLP	CEL	EDY	PGS	TCPHC	SAQ	ITT	AVD
	HVV	GKM	SWV	VCT	AIT	LAC	PCCC	IPF	LCN
	STK	DVR	TCPKCK		QAV	FVY	KIL		
DDK	FTT	LPC	ELE	DYP	GSI	TCPHC	AQI	TTA	VDH
	VVG	KMS	WVV	CTA	ITL	ACL	PCCC	PFL	CNS
	TKD	VRH	TCPKCK		AVF	VYK			
DKF	TTL	PCE	LED	YPG	TCPHC	QIT	TAV	DHV	VGK
	MSW	VVC	TAI	TLA	PCCC	FLC	NST	KDV	
	TCPKCK		VFV	YKI					

Once we have the representation method explained in section 3.3 can be used for classification.

The overview of the above 3 methods is given below:

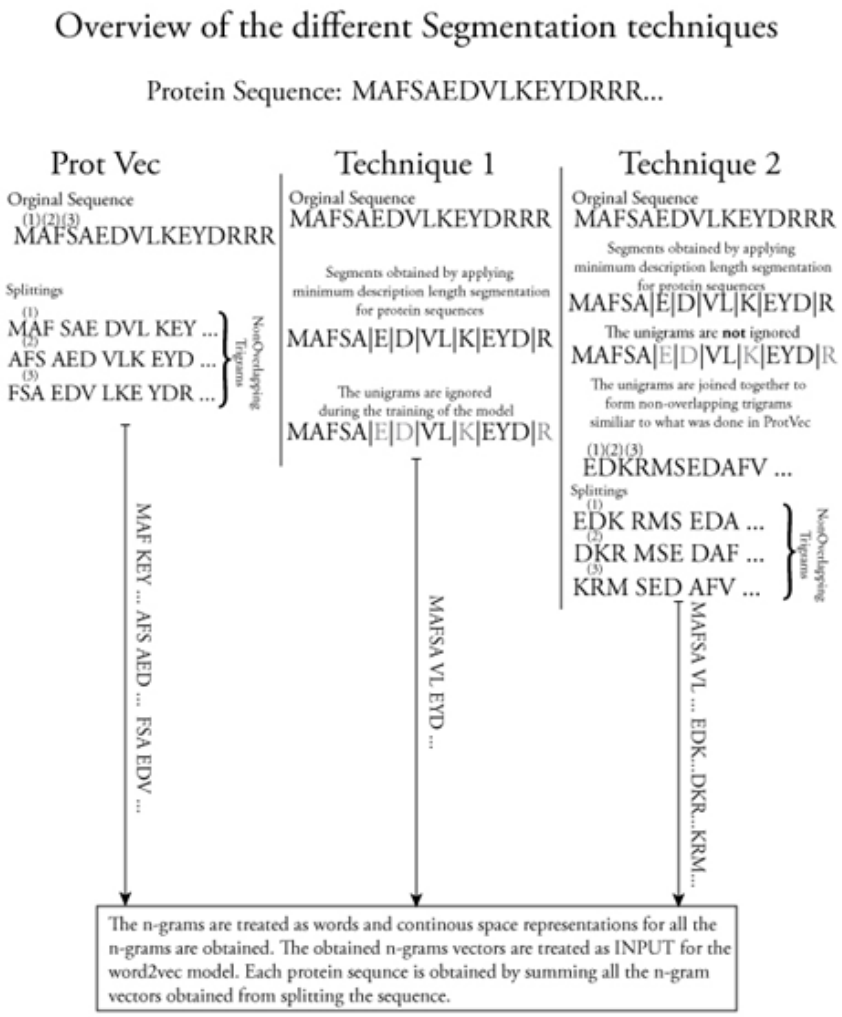

Fig. 4. Overview of the representation learning methods used.

4 Results and Discussions

S.No	Family name	# Sequences	Graph Notation
1	50S ribosome-binding GTPase	3084	1_3084
2	ATP synthase alpha/beta family, nucleotide-binding domain	2387	2_2387

3	7 transmembrane receptor (rhodopsin family)	1820	3_1820
4	Amino acid kinase family	1750	4_1750
5	Ribosomal protein S14p/S29e	997	5_997
6	Hsp90 protein	390	6_390
7	Tropomyosin	84	7_84

Table 1. Family Name and their representation in graphs along with the number if sequences in family

4.1 Specificity

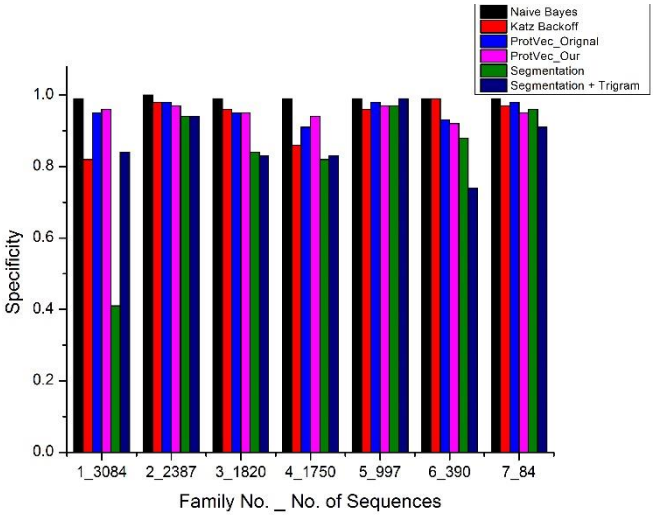

Fig. 5. The specificity i.e. true negative rate for all the mentioned approaches.

4.2 Sensitivity

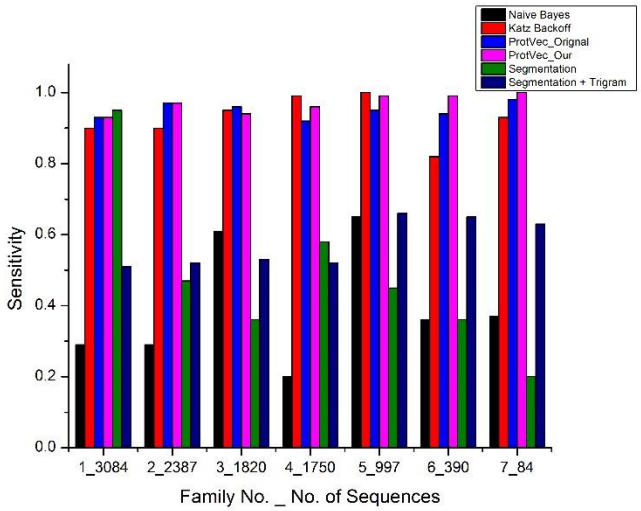

Fig. 6. The sensitivity i.e. true positive rate for all the mentioned approaches

4.3 Accuracy

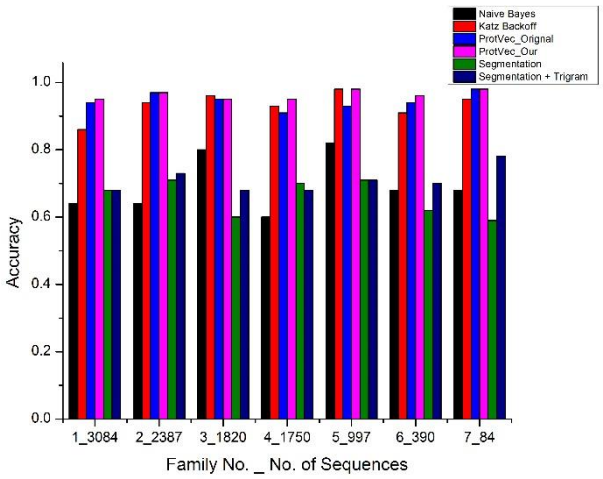

Fig. 7. The accuracy for all the mentioned approaches

Initially, we tried using standard NLP techniques. We tried Naive Bayes classifier and Katz back-off language model. Naïve Bayes didn't perform well, but Katz back-off

model was really performing well with high accuracy. This is because Katz back-off is a context-based approach whereas Naive Bayes assumes each word as independent entities. We were able to re-implement the word2vec based on overlapping trigrams in the original paper. Our results were similar to the original paper. When we extended word2vec by using segments obtained from Minimum Description length using Viterbi algorithm, we found that the performance became poorer. This is because the segments that we obtained were mostly unigrams and lead to loss of information. Then we tried a variant of segmentation in which we kept the segments of size greater than 1 as it is but we created overlapping trigrams for the unigrams in between the larger segments. This also didn't show any improvements.

We found that the number of unique overlapping trigrams was 9424 for 546290 sequences which is a very good limit for 546290 sequences. This is the reason why overlapping trigrams performed well since it captured most of the information very well. The number of unique segments was 487275 which is very high for 546790 sequences and the huge number of segments clearly indicates that it didn't capture the information well, because of the lack of sufficient amount of training data so their representation is not well learnt. When we combined segments with overlapping trigrams, the number of unique segments with trigrams went up to 829176. This is a very huge number in terms of capturing unique information. Even though the number of unique overlapping trigrams was just 9424, the number of overlapping trigrams in between segments are very high since the segments are of variable length and the trigrams formed between two different segments are really different leading to increase in the number of overlapping trigrams.

5 Conclusion

We were able to replicate the ProtVec implementation. On comparing the language model having discrete feature space in comparison to continuous feature space the latter performed better. This is because when we try to perform any classification task, the discrete features get affected by the curse of dimensionality. Whereas when we have continuous feature space we can restrict the dimensions. Representing words as unique, discrete ids furthermore leads to data sparsity, and usually means that we may need more data in order to successfully train statistical models.

One more advantage of the approach is that the representations could be learnt and saved and could be used for some other task as they are class independent.

Segmentation couldn’t perform well because the number of segments obtained were really high and their number of occurrences wasn’t enough to learn a good representation. A better solution would be to do class wise segmentation.

6 Future Work

There are few deep learning techniques that can be used to learn a better continuous representation. Convolution Neural Network can be used to get embedding of ngrams at different resolutions i.e. different values of n in a single go. Longest Short Term

Memory can be used to capture the sequential information of domains in protein which is quiet important for the protein from same family. The segmentation learnt for the sequence can be trained from the family instead of utilizing the whole corpus this might capture the domains of the protein family.

7 References

1. Asgari E, Mofrad MRK (2015) Continuous distributed representation of biological sequences for deep proteomics and genomics. PLoS One 10:1–15. doi: 10.1371/journal.pone.0141287
2. Bai J, Nie J, Paradis F (2004) Using language models for text classification. Proc Asia Inf … 1–6.
3. Jurafsky D, Martin JH (2014) N-Gram. 28.
4. Kit C, Wilks Y (1999) Unsupervised Learning of Word Boundary with Description Length Gain. Proc CoNLL99 ACL Work pages.
5. Mikolov T, Corrado G, Chen K, Dean J (2013) Efficient Estimation of Word Representations in Vector Space. Proc Int Conf Learn Represent (ICLR 2013) 1–12. doi: 10.1162/153244303322533223
6. Neduva V, Russell RB (2006) DILIMOT: Discovery of linear motifs in proteins. Nucleic Acids Res 34:350–355. doi: 10.1093/nar/gkl159
7. Robinet V (2004) MDLChunker : a MDL-based Model of Word Segmentation. 2866–2871.