Identification of Protein Language Units using Protein Oriented Tokenization Methods and Language Models

Progress Report

by

Burak Suyunu

B.S., Computer Engineering, Boğaziçi University, 2017

M.S., Computer Engineering, Boğaziçi University, 2020

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2024

ABSTRACT

Identification of Protein Language Units using Protein Oriented Tokenization Methods and Language Models Progress Report

There is a rapidly increasing discrepancy between the number of unlabeled and labeled protein sequences. Several studies have attempted to bridge this gap by utilizing semi-supervised learning for protein sequence modeling. These studies involve pre-training models with a large quantity of unlabeled data and then applying the representations to various downstream tasks. However, the majority of pre-training techniques rely exclusively on language modeling and natural language-based tokenization methods which are not necessarily the best for the protein sequences, resulting in limited performance. The most effective methods in computational biology incorporate both machine learning and evolutionary information, despite challenges such as computational expense and unavailability for certain proteins. In order to address the unique characteristics of protein sequences, this study focuses on evolutionary subword-based tokenization methods to identify protein language unit boundaries. We modify how Byte-pair Encoding chooses new tokens using pairwise sequence alignment. We propose three different methods and compare them in terms of some important statistics and linguistics laws.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Proteins play a vital role in the various biological processes that are essential for the maintenance and regulation of life. They are three-dimensional structures that can be represented in a textual form as sequences of amino acids, which play a significant role in determining their structure and function. In a similar way to natural languages, we can consider proteins to be written in a language that we refer to as the "language of life."

While natural languages can be read and understood by humans, we are not yet capable of understanding the language of life. We do not have an understanding of the vocabulary of this language, which would be the basic units of this language - similar to words in natural languages. With natural languages, we can ask native speakers to confirm that a specific structure is indeed a word and how it combines with other words to form meaningful sentences. However, when it comes to breaking a protein down into subunits, we must rely on distributional analyses, information theory measures and indirect methods to determine whether these units provide an appropriate vocabulary.

The use of textual representation of proteins has allowed for the application of natural language processing techniques in the study of proteins, resulting in major breakthroughs in tasks such as protein structure prediction. However, despite these advances, our understanding of the language of life remains at the "processing level" and we have yet to reach a deeper understanding of the language.

In natural languages, the letters of the alphabet are used to form words, the smallest units of information, and these words are then combined based on syntactic and semantic rules to create sentences. In a similar way, amino acids can be thought of as the letters in the alphabet of the "language of life", and protein sequences can be thought of as sentences [4, 5]. However, unlike natural languages, we do not yet

have a clear understanding of where the boundaries of words in protein sequences lie. Additionally, while words in natural languages are formed from sequential and non-overlapping strings of characters, this may not be the case in the language of life. Proteins exist in three-dimensional form, and protein words can also be formed from amino acids that are far apart in the protein sequence but close in 3D structure. Furthermore, there may be amino acids that are part of multiple protein words (functional units), resulting in overlapping words. Due to these significant differences between words in proteins and natural languages, we refer to the former as "protein language units" (PLUs) in this study.

1.1. Motivation, Objective and Contributions

Proteins are a vital type of molecule that can be thought of as the physical embodiment of the genetic code and play a crucial role in maintaining the processes that are essential for life [6]. These "building blocks of life" are found in nature in three-dimensional form. Interestingly, a significant amount of information about the properties of proteins, including their structures and functions, can be found in their one-dimensional sequences of amino acids [7–9].

Previous research has been based on inaccurate assumptions regarding the fundamental units of information in the language of life. These assumptions include using single amino acids, consecutive sub-sequences of amino acids, protein domains, or applying sequential tokenization approaches from NLP directly to protein sequences. However, using amino acids as the basic units of meaning limits the vocabulary, which is not sufficient given the complexity of the protein language. Protein domains, on the other hand, do not provide the ideal granularity to serve as a vocabulary for the protein language since they neglect portions of protein sequences that are not part of domains. Moreover, many proteins have only a single domain. The k-mer approach is also limited since it assumes that all language units are of equal length. Tokenization approaches in NLP, such as Byte Pair Encoding (BPE), can produce units of different lengths, but they assume that language units do not overlap and consist of a consec-

utive string of characters. This assumption is not necessarily true for the language of life.

In this progress, we developed several evolutionary subword tokenization methods where we extended the subword tokenization algorithm, BPE, by incorporating evolutionary information at the amino acid level using substitution matrices BLOSUM [10] and PAM [11]. We see it as a novel and clever way of embedding evolutionary information into protein language units. Then we examined these methods in the light of different linguistic laws.

2. BACKGROUND

Decoding the data contained in variations of protein sequences has been a persistent challenge in the field of biology. Similarly, in artificial intelligence, understanding the meaning of words based on the context in which they appear, is a problem known as natural language understanding which is also the phenomenon known as distributional hypothesis [12].

2.1. Tokenization of Protein Sequences

Tokenization involves dividing a piece of text into smaller units, such as words, phrases, symbols, or other meaningful elements, called tokens. This step is usually the initial one in any text processing pipeline, and the selection of the tokenization method can greatly influence the outcome of following NLP operations. Thus, it is important to be thoughtful about the most appropriate tokenization approach for a specific task. The principal tokenization techniques used in natural language processing are illustrated below, using the sentence "She likes playing chess" as an example. The identified tokens are separated by the dash ('-') and the space is represented with underscore ('_').

Word tokenization (She-likes-playing-chess): The most common type of tokenization is tokenization by whitespace and punctuation, or rule-based tokenization, both of which are examples of word tokenization, which can be defined as the decomposition of sentences into words. While this seems to be the most intuitive tokenization, it is not suitable for languages that are not separated by spaces, such as Chinese, Thai, Japanese, as well as the language of life.

Character tokenization (S-h-e-_-l-i-k-e-s-_-p-l-a-y-i-n-g-_-c-h-e-s-s): While character tokenization is very simple and has lower memory and time complexity due to the small size of the vocabulary, a larger number of tokens to represent a text makes

it more difficult for the model to learn long dependencies and meaningful representations. For example, it is much more difficult to learn a meaningful representation for the letter "l" than a representation for the word "like".

N-gram tokenization (n=3: She-he_-e_l-_li-lik-ike-kes-es_-s_p-_pl-pla-lay-ayi-yin-ing-ng_-g_c-ch-che-hess-ess): N-grams are n-length subsequences of a text. Here n can be any natural number and the value of n depends on the context. Character tokenization can be viewed as a special case of n-gram tokenization where n is equal to 1. The subsequences can be overlapping or non-overlapping

Subword tokenization (#She-#like-s-#play-ing-#ch-es-s)[1] : Subword tokenization methods are based on the idea that commonly used words should not be broken down into smaller subwords, while infrequent words should be divided into meaningful subparts. For example, the word "quietly" might be considered a rare word and split into "quiet" and "ly". Both "quiet" and "ly" appear more frequently as standalone subwords, and at the same time the meaning of "quietly" is preserved by the combination of "quiet" and "ly". Subword tokenization enables the model to process new words by breaking them down into familiar subwords. The most well-known algorithms for subword tokenization are Byte Pair Encoding (BPE) [13], WordPiece [14], Unigram [15], and SentencePiece [16]. Except for SentencePiece, these subword tokenizers use a pre-tokenizer (space tokenization) to divide the training data into words. BPE starts with an initial vocabulary that includes all the symbols in the dataset. Then, it repeatedly forms new symbols by merging the two most frequent symbols until the desired vocabulary size is reached. WordPiece is similar to BPE, but instead of merging symbols based on frequency, it merges symbols based on the likelihood of the training data after the new symbol is added to the vocabulary. Unlike BPE or WordPiece, Unigram starts with a large number of tokens and iteratively discards tokens to obtain a smaller vocabulary. The Unigram algorithm calculates a loss (often defined as log-likelihood) over the training data given the current vocabulary and a Unigram language model. SentencePiece treats the input as a raw input stream and

[1]The subwords starting with hashtag (#) character indicates that these subwords are the head of new words.

includes the space in the set of characters to be used, then it applies BPE or Unigram algorithms to create an appropriate vocabulary.

Even though advanced approaches in NLP such as word embeddings and transformer-based language models have been successfully adapted to model biological sequences and have yielded positive results in various downstream tasks, these efforts are still at the level of "molecular language processing" rather than "molecular language understanding". A significant limitation of these efforts is that they rely on incorrect basic assumptions about the fundamental units of meaning in the language of life. These assumptions include using single amino acids [17–21], windows of amino acids [22], protein domains [23], or applying sequential tokenization methods from NLP directly to protein sequences [24, 25].

Since there are no clearly defined word boundaries in protein sequences, a common strategy is to treat individual amino acids as tokens. However, since there are only 20 common naturally occurring amino acids, this results in a vocabulary that is too limited given the complexity and long-distance dependencies of the protein language. Using a fixed-sized window of amino acids as language units has the advantage of allowing overlapping sequences to be used as tokens, but it is not a practical approach to detecting word boundaries. Protein domains do not provide the ideal granularity to serve as tokens either, because many proteins contain a single domain and the domains do not cover the entire protein sequences. Subword tokenization methods such as Byte Pair Encoding (BPE) can be seen as a good balance between character-level and domain-level tokenization. However, these methods partition tokens into subwords based on frequency or likelihood, without taking into account lexical or semantic similarity [26]. Additionally, all of these tokenization methods assume that language units do not overlap and consist of a consecutive string of characters, which is not necessarily true for the language of life.

Figure 2.1 shows different possible tokenizations of the English sentence "She likes playing chess" and the protein sequence "MLIAMTQKAPAGNFVPTMGL". Charac-

ter and subword tokenization methods can be applied to both sequences in a similar way. However, since there is no explicit delimiter to separate the language units in protein sequence, word tokenization is only applicable to the English sentence. On the other hand, the effects of non-sequential and overlapping tokenization are demonstrated for the protein sequence, in which the fifth amino acid M is shared by the first language unit MLIAM and the second language unit MTQ, while the third language unit KAPA and the sixth language unit VP form a non-sequential language unit.

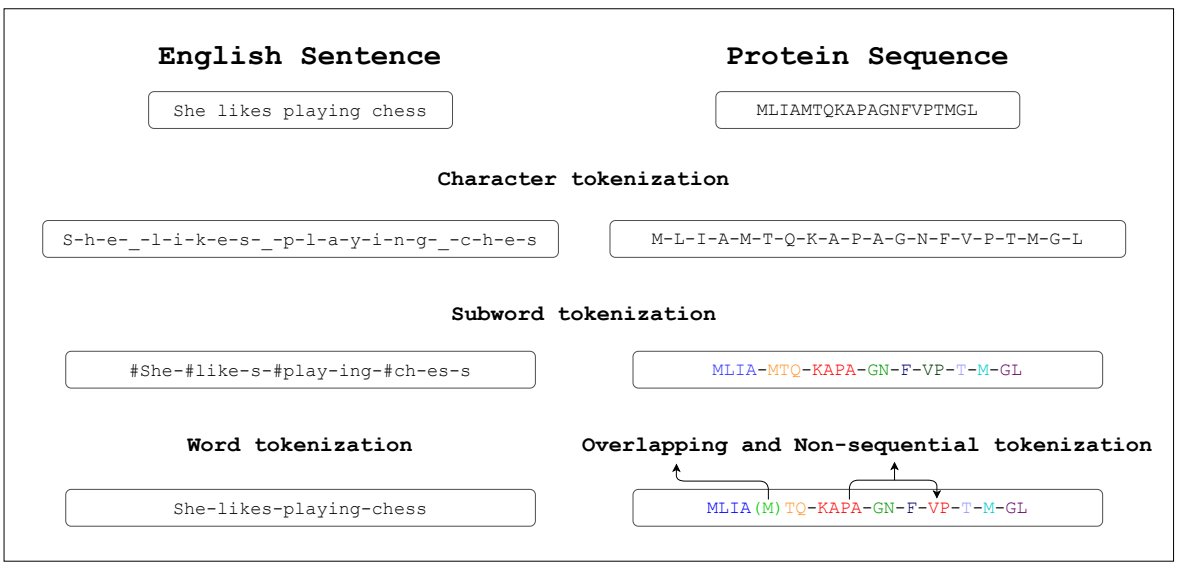


Figure 2.1: Different tokenization methods applied to the English sentence "She likes playing chess" and the protein sequence "MLIAMTQKAPAGNFVPTMGL".

3. PROPOSED WORK - EVOLUTIONARY SUBWORD TOKENIZATION

Natural languages consist of sequential terms, such as letters forming words, words forming sentences, and sentences forming documents. However, proteins occur as 3D entities, with amino acids that may be far apart in the protein sequence but close in 3D form, which allows for non-sequential combinations of amino acids, such as antigenic determinants or epitopes on protein antigens [27]. In contrast, terms in natural languages cannot be overlapping, where a syllable cannot belong to two words at the same time. However, a sequence of amino acids can be a part of two different functional units at the same time [5].

Most transformer-based pre-trained language models use character or subword tokenizers to generate vocabulary, but these tokenizers assume that tokens consist of sequential and non-overlapping units, which is not always true for proteins. These tokenizers also do not consider lexical or semantic similarity [26]. Thus, we believe that a tokenizer that can produce non-sequential and overlapping amino acid sequences, and incorporate domain-specific information, would be a better fit for the protein language.

We extend the subword tokenization algorithm, namely BPE, by incorporating evolutionary information at the amino acid level using BLOSUM [10] and PAM matrices [11]. These matrices are used as substitution matrices for sequence alignment of proteins. PAM matrices encode evolutionary approximations regarding the rates and probabilities of particular amino acid mutations, while BLOSUM encodes empirically derived substitution probabilities over a range of evolutionary periods. A detailed comparison between BLOSUM and PAM can be found in figure 3.1. Also BLOSUM62 is shown in Figure 3.2. Best methods in bioinformatics incorporate machine learning and evolutionary information via extracting and feeding the evolutionary information contained in multiple sequence alignment of proteins into machine learning models; however, compiling this information is expensive, and it is not available

for all proteins [19]. We will escape this complexity by using BLOSUM and PAM in the vocabulary generation process of BPE without introducing much computational burden.

BLOSUM	PAM
To compare closely related sequences, BLOSUM matrices with higher numbers are created.	To compare closely related sequences, PAM matrices with lower numbers are created.
BLOSUM 62 is a matrix calculated from comparisons of sequences with a pairwise identity of no more than 62%.	PAM1 is the matrix calculated from comparisons of sequences with no more than 1% divergence but corresponds to 99% sequence identity.
Based on local alignments.	Based on global alignments of closely related proteins.
BLOSUM90, BLOSUM80, BLOSUM62, BLOSUM50, BLOSUM45	PAM100, PAM120, PAM160, PAM200, PAM250

Figure 3.1: Comparison of BLOSUM and PAM.

Let's first recall how BPE works. BPE counts consecutive subwords (pairs) over the entire corpus, finds the most frequent consecutive subwords, merges them, adds the merged subword and merging rule to the vocabulary and merges all instances of the pair. BPE repeats this procedure until a target vocabulary size is reached. In our proposed methods, we will use different BLOSUM and PAM versions to alter how BPE counts and merges consecutive subwords.

Example:

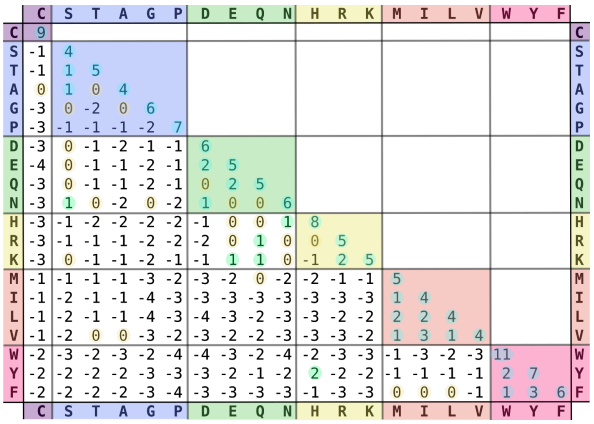
(i) **Dataset:** ('CYN', 10), ('FYN', 5), ('FYY', 12), ('RYYN', 4)

(ii) **Initialize:** ('C' 'Y' 'N', 10), ('F' 'Y' 'N', 5), ('F' 'Y' 'Y', 12), ('R' 'Y' 'Y' 'N', 4)

(iii) **Frequencies:** YN $\rightarrow$ 19, FY $\rightarrow$ 17, YY $\rightarrow$ 16, CY $\rightarrow$ 10, RY $\rightarrow$ 4

(iv) **First iter:** ('C' 'YN', 10), ('F' 'YN', 5), ('F' 'YY', 12), ('R' 'Y' 'YN', 4)

- **Dataset:** ('C' 'YN', 10), ('F' 'YN', 5), ('F' 'Y' 'Y', 12), ('R' 'Y' 'YN', 4)
- **Vocab:** ('C', 'Y', 'N', 'F', 'R', 'YN')
- **Merges:** ('Y','N'): 'YN'



	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
A	0	1	0	4																	A
G	-3	0	-2	0	6																G
P	-3	-1	-1	-1	-2	7															P
D	-3	0	-1	-2	-1	-1	6														D
E	-4	0	-1	-1	-2	-1	2	5													E
Q	-3	0	-1	-1	-2	-1	0	2	5												Q
N	-3	1	0	-2	0	-2	1	0	0	6											N
H	-3	-1	-2	-2	-2	-2	-1	0	0	1	8										H
R	-3	-1	-1	-1	-2	-2	-2	0	1	0	0	5									R
K	-3	0	-1	-1	-2	-1	-1	1	1	0	-1	2	5								K
M	-1	-1	-1	-1	-3	-2	-3	-2	0	-2	-2	-1	-1	5							M
I	-1	-2	-1	-1	-4	-3	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-1	-4	-3	-4	-3	-2	-3	-3	-2	-2	2	2	4					L
V	-1	-2	0	0	-3	-2	-3	-2	-2	-3	-3	-3	-2	1	3	1	4				V
W	-2	-3	-2	-3	-2	-4	-4	-3	-2	-4	-2	-3	-3	-1	-3	-2	-3	11			W
Y	-2	-2	-2	-2	-3	-3	-3	-2	-1	-2	2	-2	-2	-1	-1	-1	-1	2	7		Y
F	-2	-2	-2	-2	-3	-4	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	1	3	6	F
	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	

Figure 3.2: The BLOSUM62 matrix, the amino acids have been grouped and colored based on Margaret Dayhoff's classification scheme. Positive and zero values have been highlighted [1].

3.1. Method 1: Choice

Algorithm:

(i) Merge the most frequent pair.

(ii) Choose C candidate pairs from all possible pairs using pair frequencies as their choosability probabilities. (Name comes from *numpy.random.choice*())

(iii) Merge a maximum of K pairs that align with the most frequent pair above a certain alignment threshold.

Example ($C = 2, K = 1$):

(i) **Dataset:** ('CYN', 10), ('FYN', 5), ('FYY', 12), ('RYYN', 4)
Vocab: ('C', 'Y', 'N', 'F', 'R')

(ii) **Initialize:** ('C' 'Y' 'N', 10), ('F' 'Y' 'N', 5), ('F' 'Y' 'Y', 12), ('R' 'Y' 'Y' 'N', 4)

(iii) **Frequencies:** YN $\rightarrow$ 19, FY $\rightarrow$ 17, YY $\rightarrow$ 16, CY $\rightarrow$ 10, RY $\rightarrow$ 4

- Merge the most frequent pair - **YN**

(iv) **Choice:** Choose 2 candidate pairs from all pairs except 'YN' according to this probability distribution: $[0.36, 0.34, 0.21, 0.08]$

- FY → 17 and CY → 10 has chosen.

(v) **Alignment Scores (BLOSUM 62, Gap Opening: -11, Gap Extension: -1):** (YN, FY) = 1, (YN, CY) = -4

- Get the alignment scores between the most frequent pair ('Y', 'N') and the candidates. Choose the most aligned 1 candidate that has an alignment score greater than 0. (YN, FY) = 1.

(vi) **First iter:**

Dataset: ('C' 'YN', 10), ('F' 'YN', 5), ('FY' 'Y', 12), ('R' 'Y' 'YN', 4)

Vocab: ('C', 'Y', 'N', 'F', 'R', 'YN', 'FY')

Merges: ('Y','N'): 'YN', ('F', 'Y'): 'FY'

- First merge ('Y', 'N') pairs, then merge ('F', 'Y') pairs.

3.2. Method 2: Frequency

Algorithm:

(i) Merge the most frequent pair.

(ii) Choose C most frequent pairs as candidate pairs.

(iii) Merge a maximum of K pairs that align with the most frequent pair above a certain alignment threshold.

Example ($C = 2, K = 1$):

(i) **Dataset:** ('CYN', 10), ('FYN', 5), ('FYY', 12), ('RYYN', 4)

Vocab: ('C', 'Y', 'N', 'F', 'R')

(ii) **Initialize:** ('C' 'Y' 'N', 10), ('F' 'Y' 'N', 5), ('F' 'Y' 'Y', 12), ('R' 'Y' 'Y' 'N', 4)

(iii) **Frequencies:** YN → 19, FY → 17, YY → 16, CY → 10, RY → 4

- Merge the most frequent pair - **YN**

(iv) **Freq:** Choose the 2 most frequent pairs as candidates ('F', 'Y') and ('Y', 'Y')

(v) **Alignment Scores (BLOSUM 62, Gap Opening: -11, Gap Extension: -1):** (YN, FY) = 1, (YN, YY) = 5

- Get the alignment scores between the most frequent pair ('Y', 'N') and the candidates. Choose the most aligned 1 candidate that has alignment score

greater than 0. (YN, YY) = 5.

(vi) **First iter:**

Dataset: (‘C’ ‘YN’, 10), (‘F’ ‘YN’, 5), (‘F’, ‘YY’, 12), (‘R’ ‘Y’ ‘YN’, 4)

Vocab: (‘C’, ‘Y’, ’N’, ‘F’, ‘R’, ‘YN’, ‘YY’)

Merges: (‘Y’,’N’): ‘YN’, (‘Y’, ‘Y’): ‘YY’

- First merge (‘Y’, ’N’) pairs then merge (‘Y’, ‘Y’) pairs.

3.3. Method 3: Genetic

Algorithm:

(i) Merge the most frequent pair.

(ii) Apply point-wise mutation (change each aminoacid) with $M\%$ to the most frequent pair C times. Add the new mutated pairs as candidates if they appear in the dataset.

(iii) Merge a maximum of K pairs that align with the most frequent pair above a certain alignment threshold.

Example ($C = 2, K = 1$):

(i) **Dataset:** (‘CYN’, 10), (‘FYN‘, 5), (‘FYY’, 12), (‘RYYN’, 4)

Vocab: (‘C’, ‘Y’, ’N’, ‘F’, ‘R’)

(ii) **Initialize:** (‘C’ ‘Y’ ‘N’, 10), (‘F’ ‘Y’ ‘N’, 5), (‘F’ ‘Y’ ‘Y’, 12), (‘R’ ‘Y’ ‘Y’ ‘N’, 4)

(iii) **Frequencies:** YN $\rightarrow$ 19, FY $\rightarrow$ 17, YY $\rightarrow$ 16, CY $\rightarrow$ 10, RY $\rightarrow$ 4

- Merge the most frequent pair - **YN**

(iv) **Mutation:** Apply point-wise mutation with 33% to (‘Y’, ’N’) 2 times. After mutation, we have (‘R’, ’N’) and (‘Y’, ’Y’). We discard (‘R’, ’N’) because there is no (‘R’, ’N’) in the dataset.

- FY $\rightarrow$ 17 and CY $\rightarrow$ 10 has chosen.

(v) **Genetic Special Alignment Scores (BLOSUM 62, Gap Opening: -11, Gap Extension: -1):** (Y**N**, Y**Y**) = -2

- Get the alignment scores between the most frequent pair’s (‘Y’, ’N’) and the candidates’ differentiating amino acids. Here, we only get the alignment

score of N and Y, which is -2, instead of the alignment score of (YN) and (YY), which is 5. Choose the most aligned 1 candidate that has an alignment score greater than 0. We don't have any candidate satisfying this condition.

(vi) **First iter:**

Dataset: ('C' 'YN', 10), ('F' 'YN', 5), ('F', 'Y', 'Y', 12), ('R' 'Y' 'YN', 4)

Vocab: ('C', 'Y', 'N', 'F', 'R', 'YN')

Merges: ('Y','N'): 'YN'

- We only merge ('Y', 'N') pairs.

These methods differ in how they generate candidate pairs for the vocabulary. The choice method is a probabilistic method that considers all possible pairs in the dataset. While this property contains lots of potential, hyperparameters must be tuned carefully to obtain the best edge. The frequency method is the most similar method to the standard BPE. While merging the most frequent pair, the frequency of current pairs may decrease. This may cause pairs with high frequency to never reach the top. So this method gives a chance to these pairs an edge for them to enter the vocabulary. The genetic method uses the mutation idea from the well-known genetic algorithm metaheuristic to better explore the pair space. The only downside of this method for now is that, the candidate sequences must be at the same length with the most frequent pair. We see these methods as clever ways to explore token space and embed the information contained in multiple sequence alignment into protein language units.

3.4. Alternative Scoring: Frequency-Alignment Score

Instead of just using the alignment score to choose pairs from candidates, we can use both the alignment score and the pair's frequency. At the Alignment Scores part, multiply each alignment score with the summation of the frequencies of the most frequent pair and the candidate pair.

Example ($C = 2, K = 1$)**:**

(i) **Frequencies:** YN $\rightarrow$ 19, FY $\rightarrow$ 17, YY $\rightarrow$ 16, CY $\rightarrow$ 10, RY $\rightarrow$ 4

- Merge the most frequent pair - **YN**

(ii) **Freq:** Choose the 2 most frequent pairs as candidates ('F', 'Y') and ('Y', 'Y')

(iii) **Frequency Alignment Scores (BLOSUM 62, Gap Opening: -11, Gap Extension: -1):** (YN, FY) = 1, (YN, YY) = 5

- **Alignment Scores:** (YN, FY) = 1, (YN, YY) = 5.
- **Frequency Alignment Scores:** (YN, FY) = $(19+17)*1 = 36$, (YN, YY) = $(19+16)*5 = 175$.

4. EXPERIMENTS

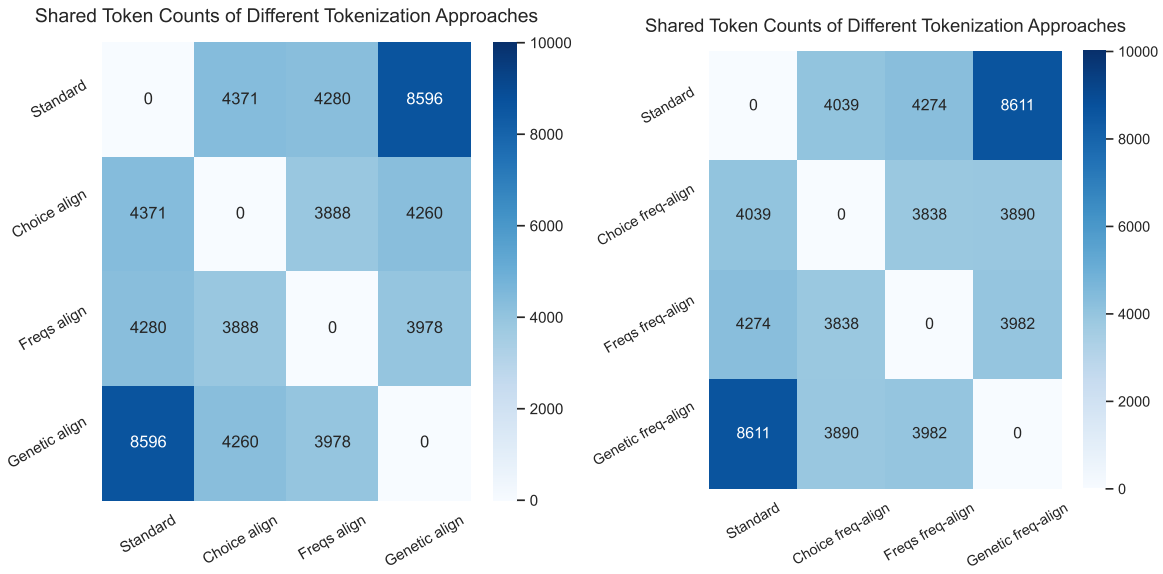
For the experiments, we used human taxonomy of the UniRef50 database that contains 68978 protein sequences. The UniRef100 database is a collection of identical protein sequences and sub-fragments that are 11 or more residues long and come from any organism. These sequences are merged into a single UniRef entry, which displays the sequence of a representative protein, the accession numbers of all the merged entries, and links to the corresponding UniProtKB and UniParc records. UniRef90 is created by clustering UniRef100 sequences that are 11 or more residues long. This is done using the MMseqs2 algorithm, which groups sequences together based on their similarity. Each cluster is made up of sequences that have at least 90% sequence identity and 80% overlap with the longest sequence in the cluster, also known as the "seed sequence." Similarly, UniRef50 is created by clustering UniRef90 seed sequences that have at least 50% sequence identity and 80% overlap with the longest sequence in the cluster. This results in a smaller set of protein sequences that are more closely related to each other [28].

We evaluated all three of the proposed methods using 10.000 as the vocabulary size, BLOSUM62 as the main substitution matrix, 100 as the candidate count, C, one extra merge, K, 33% mutation percentage for the genetic method, and both alignment and frequency alignment scoring. Also, we only applied the extra merge step for pairs that are longer than 3. As extra experiments we tried 2 extra merges, BLOSUM90 and PAM70 with the choice method. If no substitution matrix and scoring method are mentioned, you can assume that BLOSUM62 and alignment scoring are used.

All the methods and hyperparameters are evaluated under their shared token counts, average token counts, and their behavior under zipf's law, brevity law, Heaps' law, and Menzerath's law.

4.1. Shared Token Counts

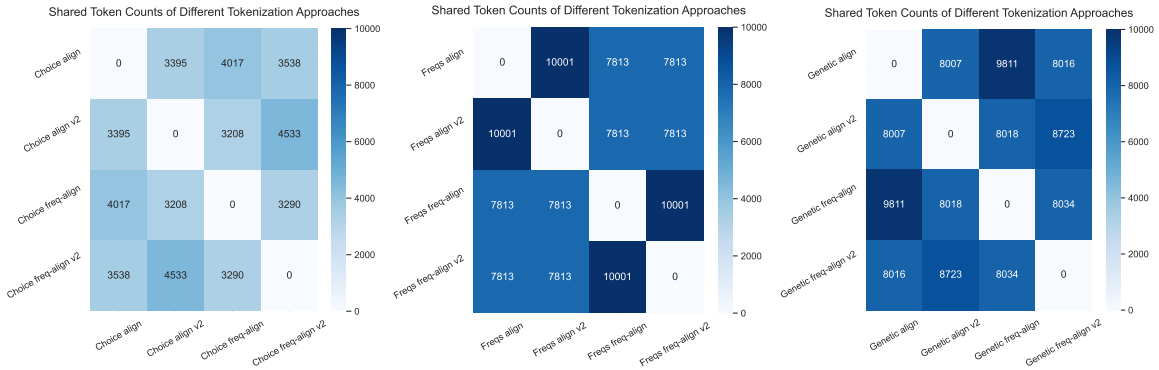
In this experiment, we calculated the number of shared tokens between different methods to observe how far these methods ended up from each other in a naive way. Figure 4.1 shows that choice and frequency methods generated quite many different tokens than the standard BPE. Whereas the genetic method ended up with a similar vocabulary to the standard BPE which signals something went wrong with the genetic method. There could be two reasons: First, the insufficient number of candidates; second, alignment score calculations. For the first one we can increase the candidate count, for the second we can calculate the alignment score the regular way but then every alignment will get score greater than 0 which is not optimal. Instead we can mutate according to the substitution matrix instead of fully randomly.



(a) Alignment Score (b) Frequency Alignment Score

Figure 4.1: Shared token counts of standard BPE, choice, frequency, genetic methods using alignment score and frequency alignment score in (a) and (b), respectively.

Figure 4.2 shows how methods vocabulary changes when we repeat the experiment. While the vocabulary of the choice method changes significantly; there is no change in the frequency method because of its deterministic nature. Genetic method has little change possibly for the reasons we mentioned above. Figure 4.3 shows comparisons of other variations for the choice method, but there isn't much to infer.



(a) Choice methods (b) Frequency methods (c) Genetic methods

Figure 4.2: Shared token counts of (a) the choice method, (b) the frequency method and (c) the genetic method with different and repeated hyperparameters.

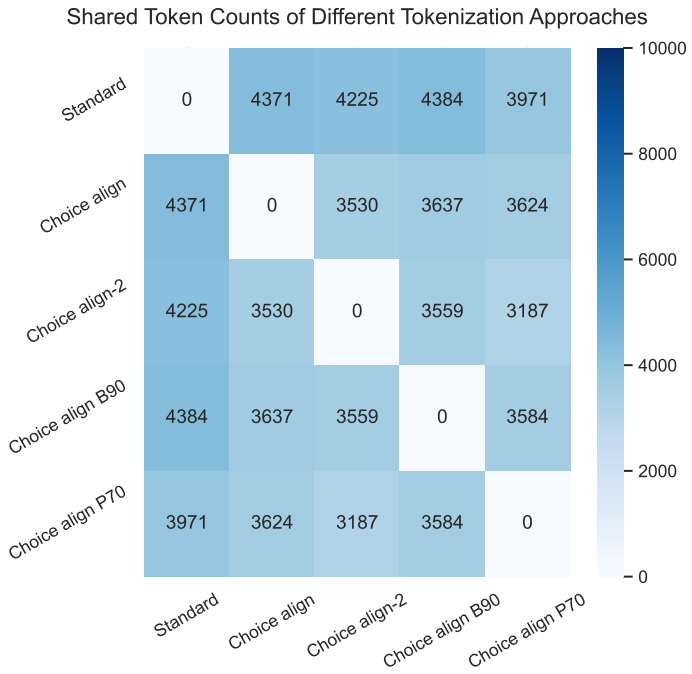


Figure 4.3: The choice method with different hyperparameters.

4.2. Average Token Length

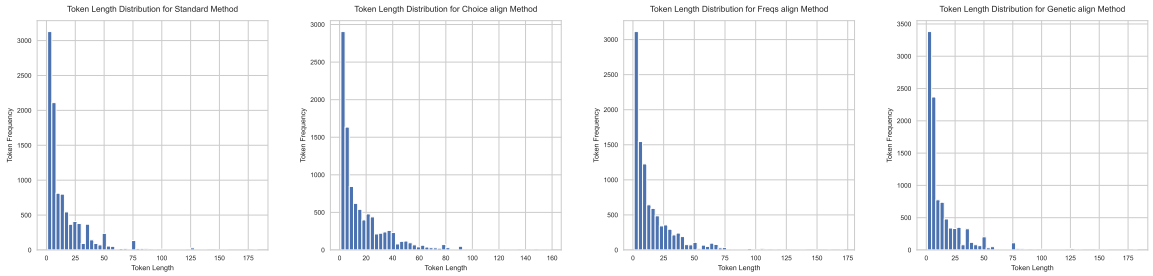
BPE starts with a vocabulary of single characters and generates longer tokens in time. Having a longer average token length can indicate that the method found tokens before its time has come, which is good. The average token lengths of different tokenization methods are as follows:

- **Standard BPE:** 15.49
- **Choice method with Alignment scoring:** 17.37
- **Choice method with Alignment scoring and 2 extra merges:** 18.49
- **Choice method with Frequency-Alignment scoring:** 16.99
- **Frequency method with Alignment scoring:** 15.60
- **Frequency method with Frequency-Alignment scoring:** 15.58
- **Genetic method with Alignment scoring:** 14.01
- **Genetic method with Frequency-Alignment scoring:** 14.01
- **Choice method with Alignment scoring and BLOSUM90:** 17.53
- **Choice method with Alignment scoring and PAM70:** 17.81

Here again, the choice method shines brighter than other methods, where it has the highest average token length. Also adding an extra merge option to the choice method also increased the average token length. This indicates that the choice method finds longer tokens related to the most frequent token rather than shorter ones. We can see the difference in their token length distributions in Figure 4.4.

4.3. Zipf's Law

Zipf's Law is a statistical principle that describes the distribution of frequencies of elements in a dataset. It is named after the linguist George Zipf, who first observed this pattern in the distribution of word frequencies in natural languages. Zipf's Law is



(a) Standard BPE (b) Choice Method (c) Frequency Method (d) Genetic Method

Figure 4.4: Token length distributions of the (a) standard BPE, (b) choice, (c) frequency and (d) genetic methods.

a power-law distribution and is often expressed mathematically as:

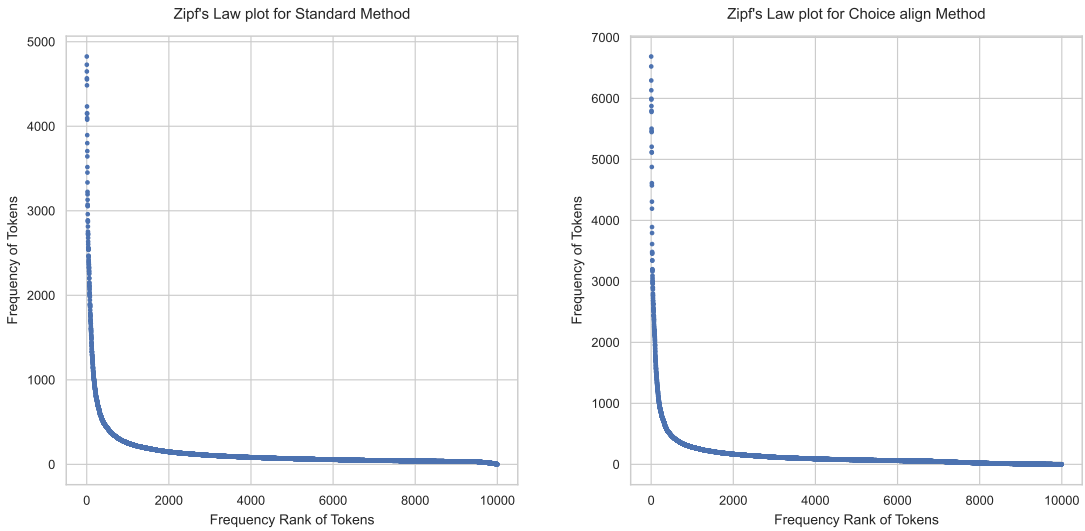
$$f(r) = \frac{1}{r^{\alpha}}$$

where:

- $f(r)$ is the frequency of the element ranked at
- α is a constant typically close to 1, and
- r is the rank of the element.

In simpler terms, Zipf's Law states that the frequency of a particular element is inversely proportional to its rank. This means that a few elements occur very frequently, while the majority of elements occur infrequently. The law is often applied to the distribution of words in texts, where a small number of words (like common words such as "the," "and," etc.) are used frequently, while the majority of words are used less frequently. Figure 4.5 shows how tokenization methods comply with the law.

But a better than the vanilla Zipf's plot is its log-log version. In Zipf log-log plot, we expect to have a slope of -1 for natural languages. The slope of log-log plots of Zipf's law for different methods are as follows:

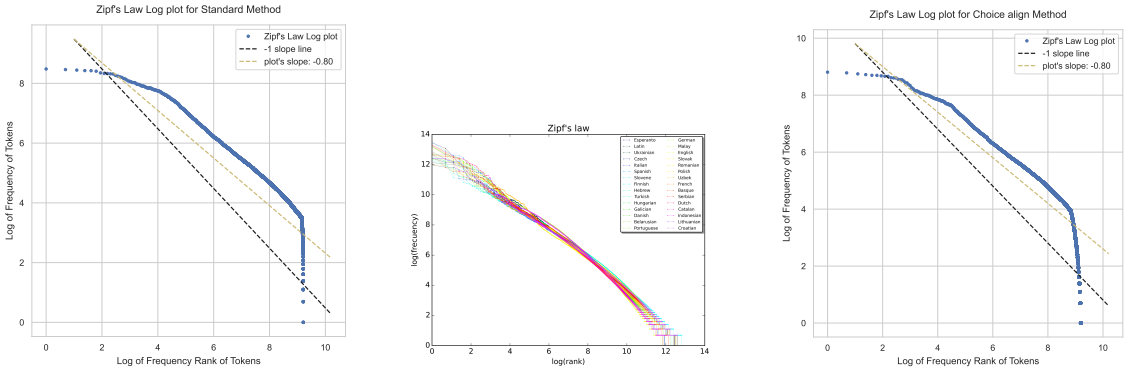


(a) Standard BPE (b) Choice Method

Figure 4.5: Zipf's law plot (frequency rank of tokens vs frequency of tokens) for (a) the standard BPE and (b) the choice method.

- **Standard BPE:** -0.80
- **Choice method with Alignment scoring:** -0.80
- **Choice method with Alignment scoring and 2 extra merges:** -0.80
- **Choice method with Frequency-Alignment scoring:** -0.79
- **Frequency method with Alignment scoring:** -0.78
- **Frequency method with Frequency-Alignment scoring:** -0.78
- **Genetic method with Alignment scoring:** -0.80
- **Genetic method with Frequency-Alignment scoring:** -0.80
- **Choice method with Alignment scoring and BLOSUM90:** -0.79
- **Choice method with Alignment scoring and PAM70:** -0.79

Most of the methods converge to -0.8, which is not far from -1, but still, it is not -1. This may be an indication of that the proteins do not work exactly like natural languages. Figure 4.6 shows the log-log plots of the standard BPE, the choice method, and some Wikipedia articles.



(a) Standard BPE (b) Wikipedia (c) Choice Method

Figure 4.6: Zipf's law plot (log of frequency rank of tokens vs log of frequency of tokens) for (a) the standard bpe, (b) the first 10 million words in 30 Wikipedias (as of October 2015) [2] and (c) the choice method in a log-log scale.

4.4. Brevity Law

Brevity Law (or Zipf's Law of Abbreviation) is a concept related to the distribution of abbreviations or acronyms in a language. It is an extension of Zipf's Law, which describes the distribution of frequencies for elements in a dataset, such as words in a natural language.

In the context of abbreviations, Zipf's Law of Abbreviation suggests that a small number of abbreviations are used very frequently, while a large number of abbreviations are used infrequently. In other words, Brevity Law qualitatively states that the more frequently a word is used, the shorter that word tends to be, and vice versa; the less frequently a word is used, the longer it tends to be.

Figure 4.7 shows how Brown Corpus fits Brevity Law where we observe a decrease in the word count as the word length increases. At the same time, we can observe words from every length for every order of word counts. We can observe the negative slope for different tokenization methods in Figure 4.8; whereas we can only observe tokens from every length for every order of token counts for the choice method. This trend continues for different versions of the choice method in Figure 4.9. The sharp cut-

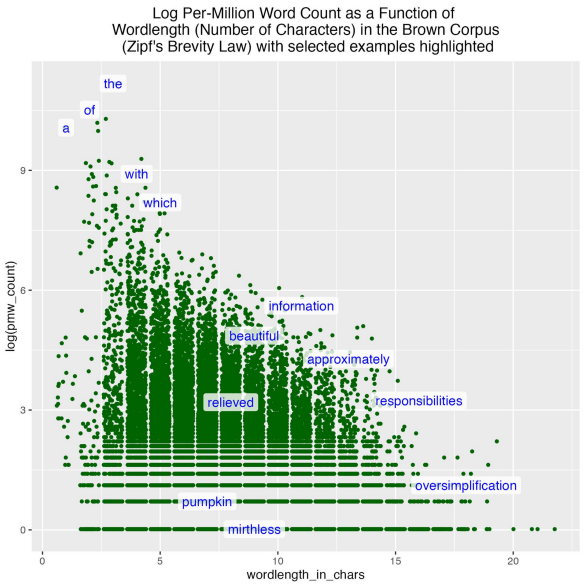
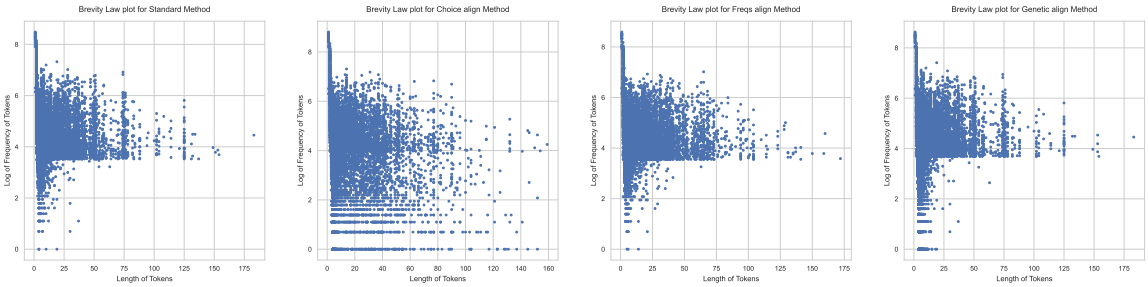


Figure 4.7: Log per-million word count as a function of word length (number of characters) in the Brown Corpus, illustrating Brevity Law [3].

off for longer sequences for methods other than the choice method is actually really interesting, which indicates that the standard BPE can not generate longer tokens that appear rarely in the documents. The shorter tokens that appear relatively less in the documents are the tokens generated in the early phases of the BPE algorithm. At some point, BPE unwittingly creates a sharp threshold on the count of tokens to be accepted for the vocabulary. This is also counter-intuitive for a natural language. Even though we use longer words rarely, we need those rare words to embellish our lives. So, we can safely say that the choice method takes the standard BPE closer to being more natural.

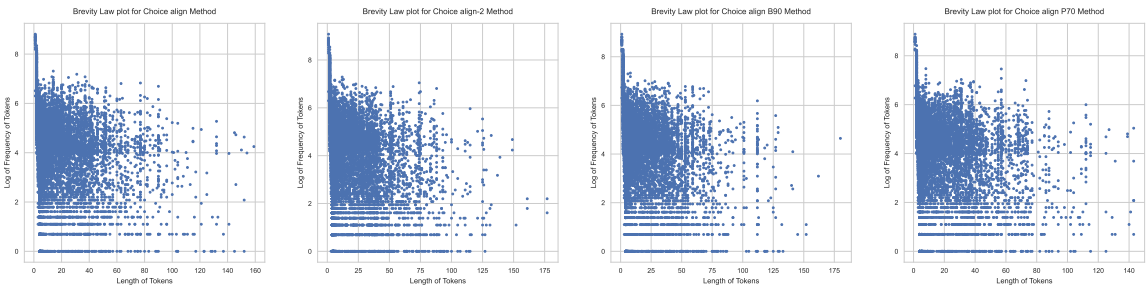
4.5. Heap's Law

Heap's Law, named after Stuart Heap, is an empirical formula that describes the relationship between the size of a vocabulary (the number of distinct words) in a document or a collection of documents and the total number of words. Heap's Law is commonly applied in natural language processing and information retrieval.



(a) Standard BPE (b) Choice Method (c) Freq Method (d) Genetic Method

Figure 4.8: Brevity Law plot (length of tokens vs log of frequency of token counts) for the standard BPE, choice, frequency, genetic methods.



(a) Choice Method (b) Choice Method with 2 extra merge (c) Choice Method with BLOSUM90 (d) Choice Method with PAM70

Figure 4.9: Brevity Law plot (length of tokens vs log of frequency of token counts) for different versions of the choice method.

The formula is expressed as:

$$V(n) = K \cdot n^{\beta}$$

where:

- $V(n)$ is the estimated vocabulary size when the document or collection contains n words,
- K is a constant, typically in the range of 10 to 100.
- β is an exponent, typically in the range of 0.4 to 0.6.

In simpler terms, Heap's Law suggests that as the size of a document or dataset increases, the vocabulary size (the number of unique words) also increases, but at a decreasing rate. The exponent β determines the rate of growth. The larger β is, the slower the vocabulary size grows concerning the document size. Heap's Law is often used to estimate the vocabulary size needed for information retrieval systems or to assess the richness of vocabulary in a given text. It highlights the observation that as more text is added, new words continue to be introduced, but the rate of introduction decreases over time. Figure 4.10 shows how a natural language fits Heaps' law.

As plotted in Figure 4.11, the standard BPE and our proposed methods have sharper curves than expected where $K = 3300$ and $\beta = 0.08$. This can be an outcome of our small and correlated dataset, which only includes human proteins.

4.6. Menzerath's Law

Menzerath's law, also known as Menzerath–Altmann law or Menzerath's law of linguistic organization, is an empirical linguistic principle that describes the relationship between the size of linguistic constructs (such as words, phrases, or clauses) and the size of their constituents (subparts). The law is often expressed as follows: "The larger the whole, the smaller the constituents." For instance, the longer a protein (in

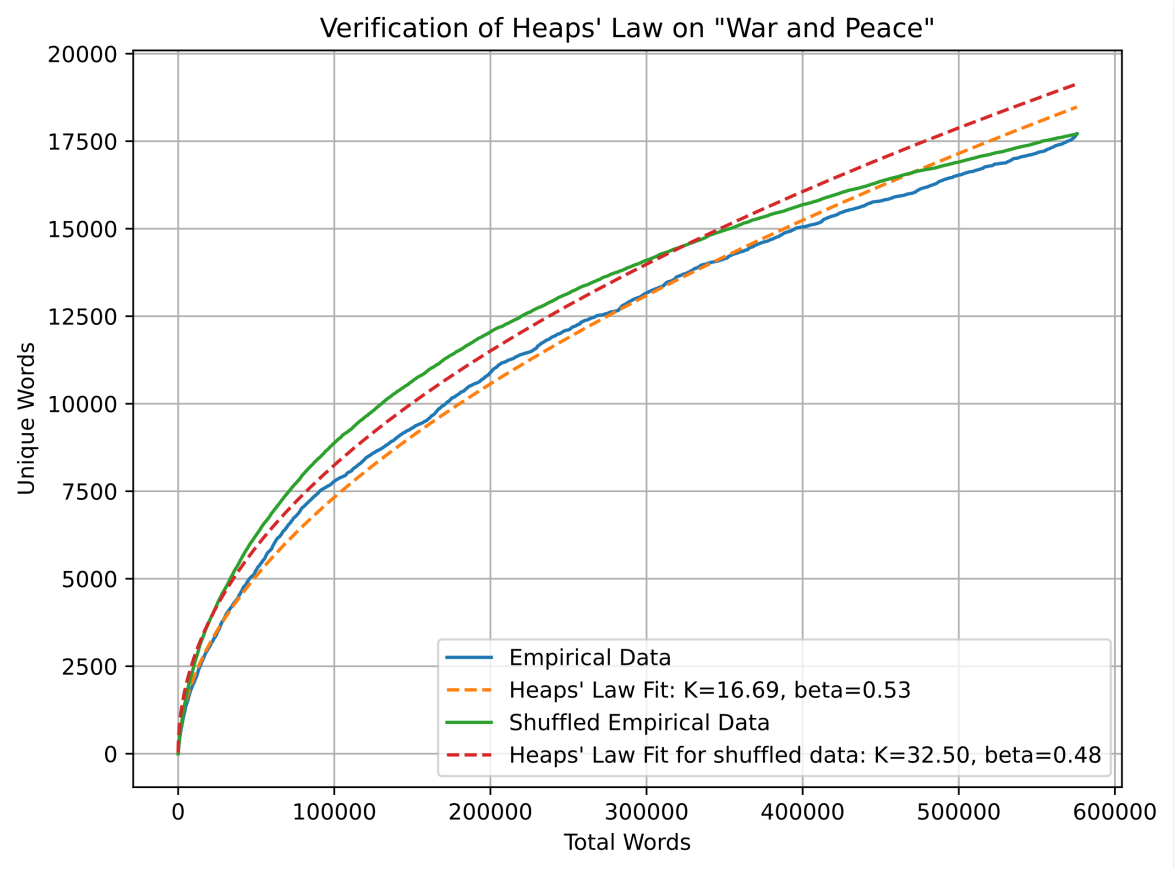
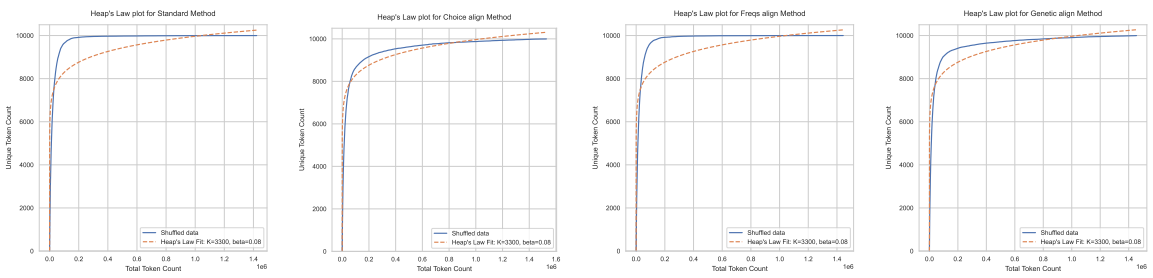


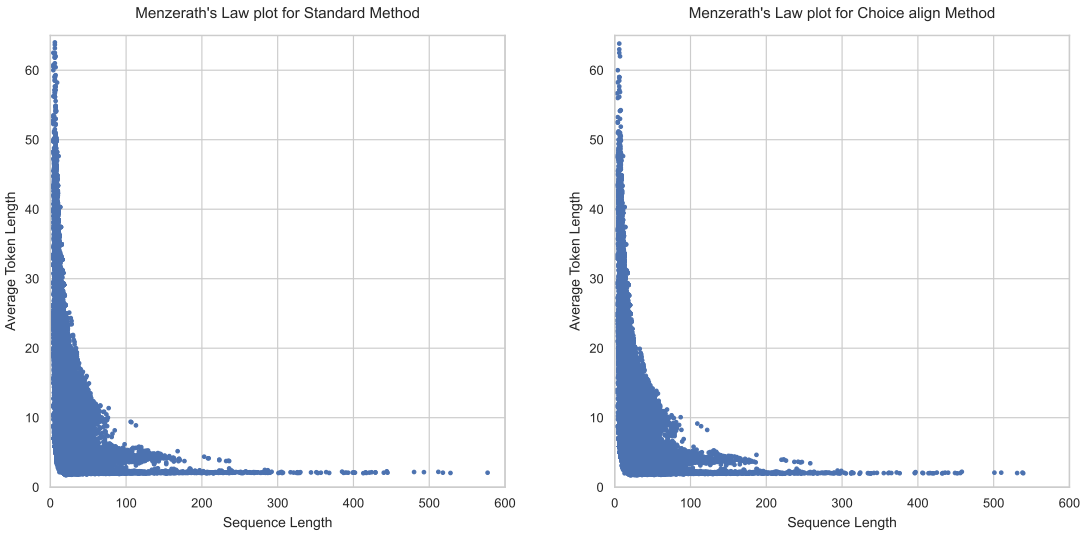
Figure 4.10: Verification of Heaps' law on War and Peace, as well as a randomly shuffled version of it. Both cases fit well to the Heaps' law with very similar exponents β, but different K



(a) Standard BPE (b) Choice Method (c) Freq Method (d) Genetic Method

Figure 4.11: Heap's Law plot (total token count vs unique token count) for the standard BPE, choice, frequency, genetic methods.

tokens), the shorter its tokens (in amino acids). Figure 4.12 shows that standard BPE and the choice method fit the law. Other methods also give a very similar plot.



(a) Standard BPE (b) Choice Method

Figure 4.12: Menzerath's Law plot (sequence length in tokens vs average token length) for the standard BPE and the choice method.

5. CONCLUSION AND FUTURE WORK

We developed three different extensions for BPE utilizing pairwise sequence alignment with BLOSUM and PAM to generate candidate pairs as alternatives to the most frequent pair chosen by the standard BPE algorithm. Then, we evaluated these methods according to some important statistics and linguistic laws. The experiments showed that the proposed choice method improved the standard BPE in terms of the fitness to the linguistic laws.

Here are the things that I plan to do in the next six months:

- Optimize the code to run faster. At the moment, it takes 10 hours to tokenize 70k protein to create a vocabulary of size 10k.
- Make more experiments; try different hyperparameters. We need to try different substitution matrices, vocabulary sizes, and candidate sizes to find the optimal combination and to test the methods for robustness.
- Implement the Evolutionary Subword Tokenization approaches to WordPiece and Unigram.
- Compare the generated tokens with the known Protein Domains and Motifs.
- Train Protein Language Models with tokens generated from the Evolutionary Subword Tokenization algorithm. Then finetune these PLMs for different downstream tasks to be able to compare the tokenization methods quantitatively.
- Start working on Graph-based Overlapping and Non-sequential Tokenization.

REFERENCES

1. Ppgardne, *Blosum62 Dayhoff Ordering*, 2022, `https://commons.wikimedia.org/w/index.php?curid=119457674`, accessed in January 2023.

2. Wikipedia, *Zipf's Law*, 2024, `https://en.wikipedia.org/wiki/Zipf's_law`, accessed in January 2024.

3. Wikipedia, *Brevity Law*, 2024, `https://en.wikipedia.org/wiki/Brevity_law`, accessed in January 2024.

4. Gimona, M., "Protein linguistics—a grammar for modular protein assembly?", *Nature Reviews Molecular Cell Biology*, Vol. 7, No. 1, pp. 68–73, 2006.

5. Ofer, D., N. Brandes and M. Linial, "The language of proteins: NLP, machine learning & protein sequences", *Computational and Structural Biotechnology Journal*, Vol. 19, pp. 1750–1758, 2021.

6. Cooper, G. M., R. E. Hausman and R. E. Hausman, *The cell: a molecular approach*, Vol. 4, ASM press Washington, DC, 2007.

7. Anfinsen, C. B., "Studies on the principles that govern the folding of protein chains", *Les Prix Nobel en 1972*, pp. 103–119, 1971.

8. Anfinsen, C. B., "Principles that govern the folding of protein chains", *Science*, Vol. 181, No. 4096, pp. 223–230, 1973.

9. Ptitsyn, O., "How does protein synthesis give rise to the 3D-structure?", *FEBS letters*, Vol. 285, No. 2, pp. 176–181, 1991.

10. Henikoff, S. and J. G. Henikoff, "Amino acid substitution matrices from protein

blocks.", *Proceedings of the National Academy of Sciences*, Vol. 89, No. 22, pp. 10915–10919, 1992.

11. Dayhoff, M. O., "A model of evolutionary change in proteins", *Atlas of protein sequence and structure*, Vol. 5, pp. 89–99, 1972.

12. Harris, Z. S., "Distributional structure", *Word*, Vol. 10, No. 2-3, pp. 146–162, 1954.

13. Sennrich, R., B. Haddow and A. Birch, "Neural machine translation of rare words with subword units", *arXiv preprint arXiv:1508.07909*, 2015.

14. Wu, Y., M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation", *arXiv preprint arXiv:1609.08144*, 2016.

15. Kudo, T., "Subword regularization: Improving neural network translation models with multiple subword candidates", *arXiv preprint arXiv:1804.10959*, 2018.

16. Kudo, T. and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing", *arXiv preprint arXiv:1808.06226*, 2018.

17. Ingraham, J., V. Garg, R. Barzilay and T. Jaakkola, "Generative models for graph-based protein design", *Advances in neural information processing systems*, Vol. 32, 2019.

18. Heinzinger, M., A. Elnaggar, Y. Wang, C. Dallago, D. Nechaev, F. Matthes and B. Rost, "Modeling aspects of the language of life through transfer-learning protein sequences", *BMC bioinformatics*, Vol. 20, No. 1, pp. 1–17, 2019.

19. Elnaggar, A., M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger *et al.*, "ProtTrans: towards cracking the

language of Life's code through self-supervised deep learning and high performance computing", *arXiv preprint arXiv:2007.06225*, 2020.

20. Min, S., S. Park, S. Kim, H.-S. Choi, B. Lee and S. Yoon, "Pre-training of deep bidirectional protein sequence representations with structural information", *IEEE Access*, Vol. 9, pp. 123912–123926, 2021.

21. Rives, A., J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma *et al.*, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences", *Proceedings of the National Academy of Sciences*, Vol. 118, No. 15, p. e2016239118, 2021.

22. Asgari, E. and M. R. Mofrad, "Continuous distributed representation of biological sequences for deep proteomics and genomics", *PloS one*, Vol. 10, No. 11, p. e0141287, 2015.

23. Coin, L., A. Bateman and R. Durbin, "Enhanced protein domain discovery by using language modeling techniques from speech recognition", *Proceedings of the National Academy of Sciences*, Vol. 100, No. 8, pp. 4516–4520, 2003.

24. Asgari, E., A. C. McHardy and M. R. Mofrad, "Probabilistic variable-length segmentation of protein sequences for discriminative motif discovery (DiMotif) and sequence embedding (ProtVecX)", *Scientific reports*, Vol. 9, No. 1, pp. 1–16, 2019.

25. Filipavicius, M., M. Manica, J. Cadow and M. R. Martinez, "Pre-training protein language models with label-agnostic binding pairs enhances performance in downstream tasks", *arXiv preprint arXiv:2012.03084*, 2020.

26. Tay, Y., V. Q. Tran, S. Ruder, J. Gupta, H. W. Chung, D. Bahri, Z. Qin, S. Baumgartner, C. Yu and D. Metzler, "Charformer: Fast character transformers via gradient-based subword tokenization", *arXiv preprint arXiv:2106.12672*, 2021.

27. Khan, F. H., *The elements of immunology*, Pearson Education India, 2009.

28. Consortium, T. U., "UniProt: the universal protein knowledgebase in 2021", *Nucleic Acids Research*, Vol. 49, No. D1, pp. D480–D489, 11 2020, https://doi.org/10.1093/nar/gkaa1100.