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

PhD Progress

Burak Suyunu - 16.01.2024

LifeLU

HORIZON

Call: ERC-2022-COG
(Call for Proposals for ERC Consolidator Grant)

Topic: ERC-2022-COG

Type of Action: HORIZON-ERC
(HORIZON ERC Grants)

Proposal number: 101089287

Proposal acronym: LifeLU

Arzucan Özgür, Gökçe Uludoğan, Enes Taylan

Introduction

- Proteins play a vital role in the maintenance and regulation of life.
- Sequences of amino acids (2D) —> determining structure and function of protein.
- Proteins as written language —> Language of Life
  - Characters —> Amino acids
  - Sentences —> Protein sequences
- No one can understand, no vocabulary, no words.

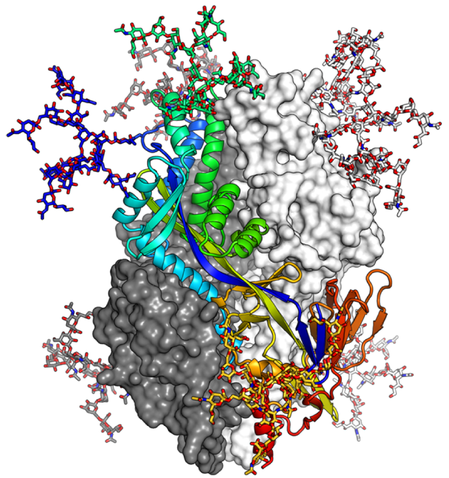


Image courtesy of Nat Commun 8, 1528 (2017)

Motivation

- Increase in unlabeled data
- Pre-training language models with a large quantity of unlabeled data
- Using tokenization methods from NLP which are not necessarily the best for the protein sequences
- The most effective methods in computational biology incorporate both machine learning and evolutionary information

Objective

- Advance beyond current language processing methods and explore new ways to identify the language units of proteins.
- Establish a new field called molecular language identification, which aims to uncover the meaning behind molecular sequences
- Provide a novel perspective on identifying protein language units.
- Develop algorithms that do not assume protein language units are sequential and non-overlapping.

Contributions

Evolutionary Subword tokenization

- Subword tokenization algorithms will be extended by incorporating evolutionary information at the amino acid level using substitution matrices BLOSUM and PAM.

Graph-based overlapping and non-sequential tokenization

- A graph-based tokenization approach that is able to identify non-sequential and overlapping protein language units will be developed.

Pre-training and fine-tuning protein language models

- Transformer-based protein language models with datasets that are tokenized with our proposed tokenization methods above will be pre-trained and fine-tuned.

End-to-end tokenization

- An end-to-end comprehensive method for recognizing tokens by training both the tokenization process and a transformer-based language model simultaneously, rather than separately will be developed.

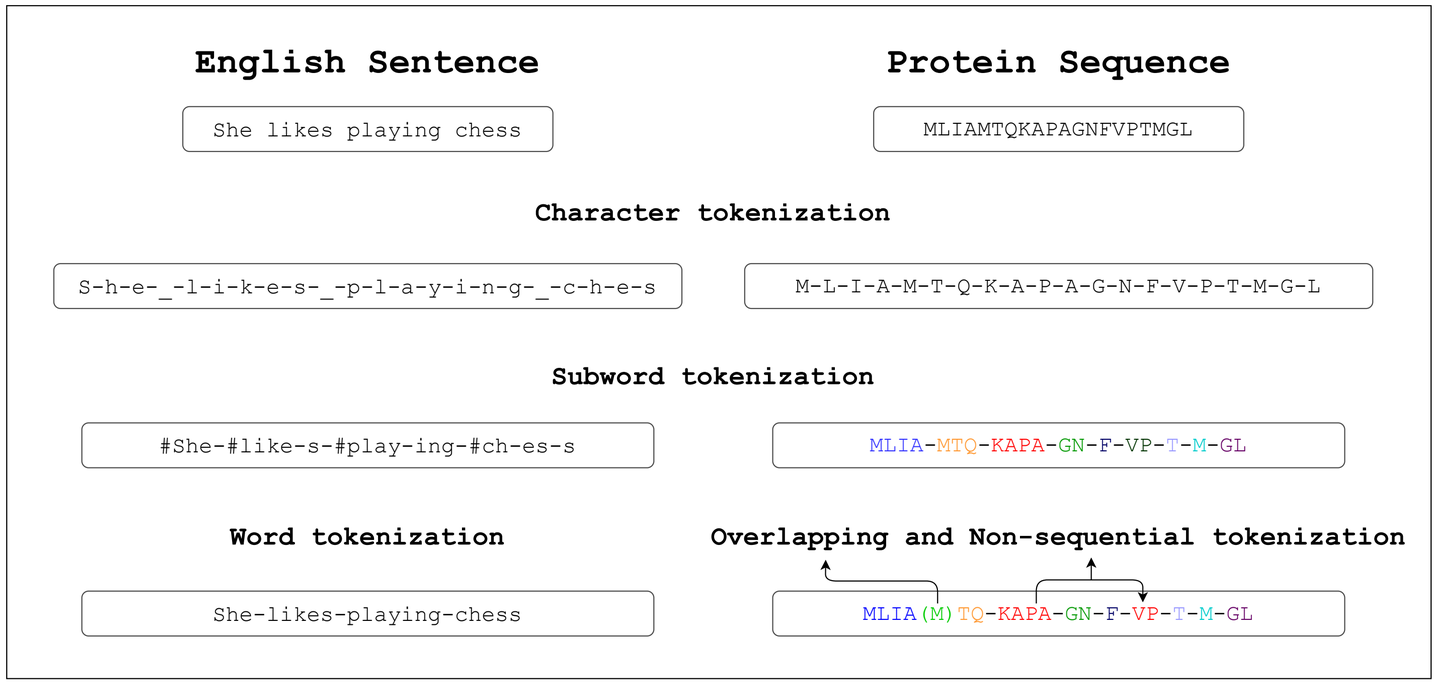
Novelty & Impact

- The goal is to open a path from **“protein language processing”** to **“protein language understanding”**
  - Relationship between sequence and function
  - De novo protein design
- Biology is a **natural science**, where accurate prediction is not always the most important aspect, but **exploration and in-depth understanding** are at least as crucial.
- Incorporating **domain knowledge** into the tokenization process will be novel
  - More efficient language model pre-training with better **compression** of input data.
  - Improve the **quality** of the final embeddings and the accuracy of downstream tasks.

Literature Review

Tokenization

- Tokenization involves dividing a piece of text into smaller units, such as words, phrases, symbols, or other meaningful elements, called tokens.



Literature Review

Tokenization (why)

- Still at the level of “molecular language processing” rather than “molecular language understanding”.
  - **Incorrect basic assumptions** about the fundamental units of meaning in the language of life - no clearly defined word boundaries.
  - **Single amino acids:** Only 20 common naturally occurring aa —> too limited.
  - **Windows of amino acids:** Allows overlapping but not for detecting word boundaries.
  - **Protein domains:** Not ideal granularity, do not cover the entire protein sequences.
  - **Sequential tokenization methods:** A good balance between character-level and domain-level tokenization. Based on frequency or likelihood which works for human languages but not certain for protein sequences.
- All assume that PLUs do not overlap and consist of a consecutive string of characters, which is not necessarily true for the language of life.

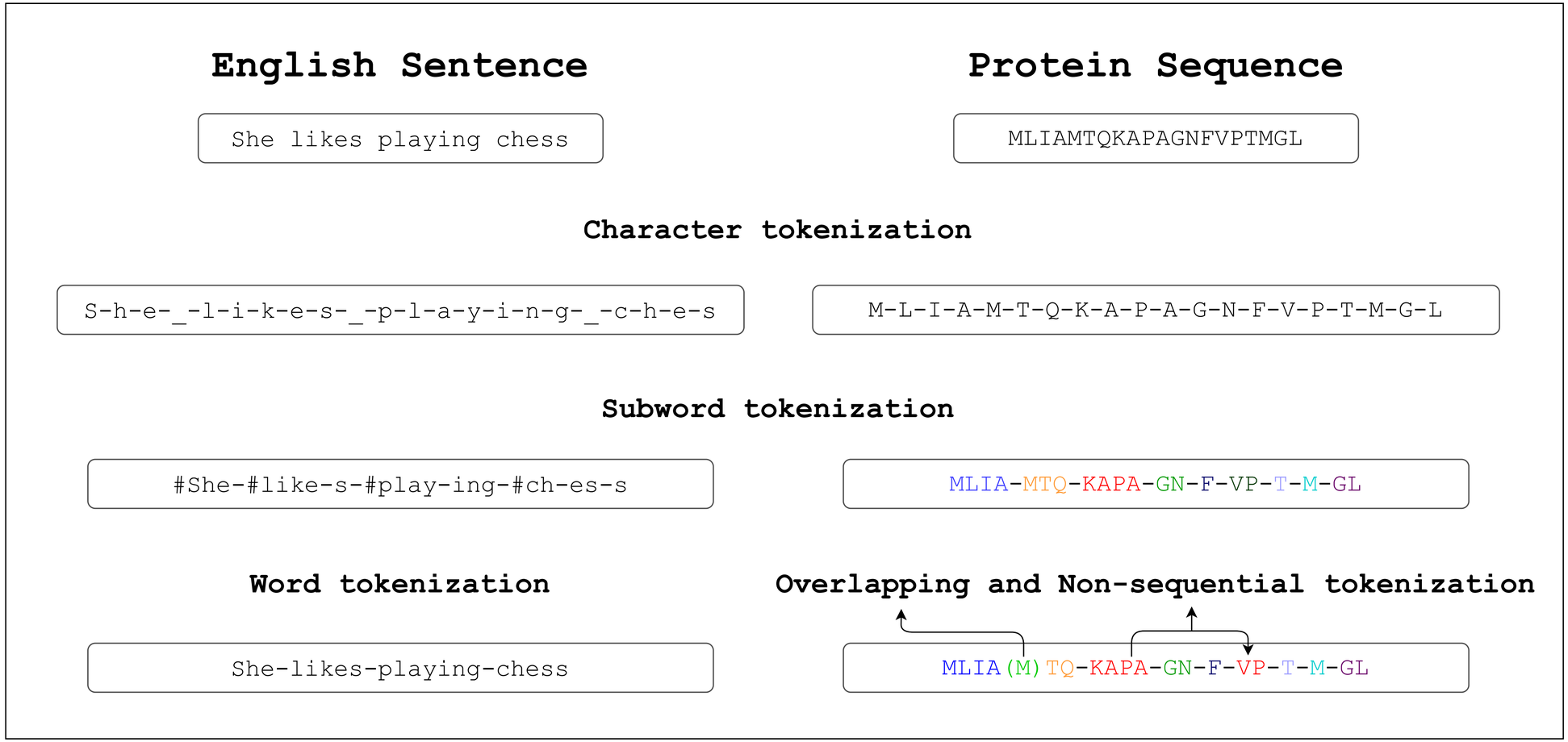
Literature Review

Subword Tokenization

- 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.
- 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), WordPiece, Unigram, and SentencePiece.
    - **BPE:** Initial vocabulary of symbols, merge two most frequent pair of symbols until desired vocabulary size.
    - **WordPiece:** Similar to BPE, likelihood instead of frequency.
    - **Unigram:** Big initial vocab, discard tokens according to loss and unigram language model
    - **SentencePiece:** More of an implementation of BPE and Unigram. Can work on text without space.

Literature Review

Tokenization

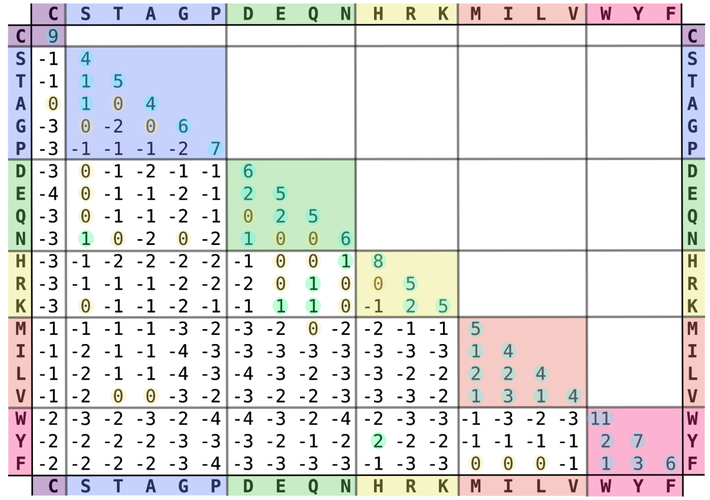


Proposed Work

Evolutionary Subword Tokenization

Approach

- Extend **subword tokenization** algorithms, namely BPE, WordPiece and Unigram, by incorporating evolutionary information at the amino acid level using **BLOSUM** and **PAM** matrices.
    - These matrices are used as substitution matrices for sequence alignment of proteins.
- Best methods in bioinformatics incorporate machine learning and evolutionary information —> Expensive
- Escape this complexity by using BLOSUM and PAM in the vocabulary generation (reduction) process of BPE and WordPiece (Unigram).



	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	

BLOSUM vs PAM

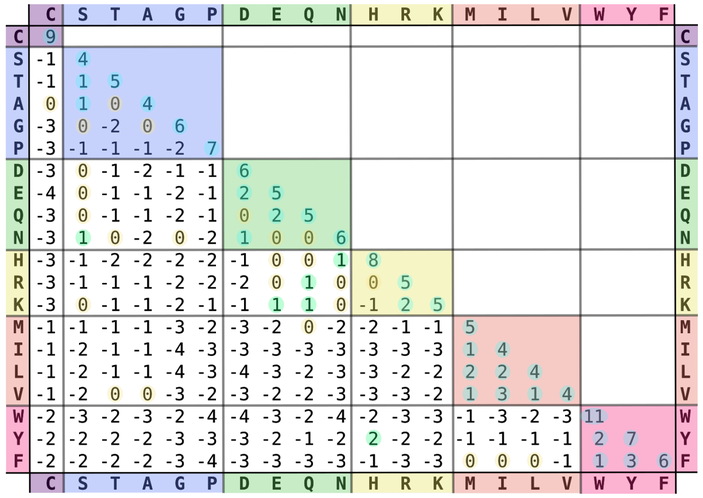
BLOSUM	PAM
To compare closely related sequences, BLOSUM matrices with higher numbers are created.	To compare closely related sequences, PAM matrices with lower numbers are created.
BLOSUM 62 is a matrix calculated from comparisons of sequences with a pairwise identity of no more than 62%.	PAM1 is the 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

Pairwise Sequence Alignment

P1:	M	A	A	Q	P
P2:	A	A	V	P	

P1:	M	A	A	Q	P
	5	4	0	-1	-11
P2:	M	A	V	P	-

Score= -3



	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	

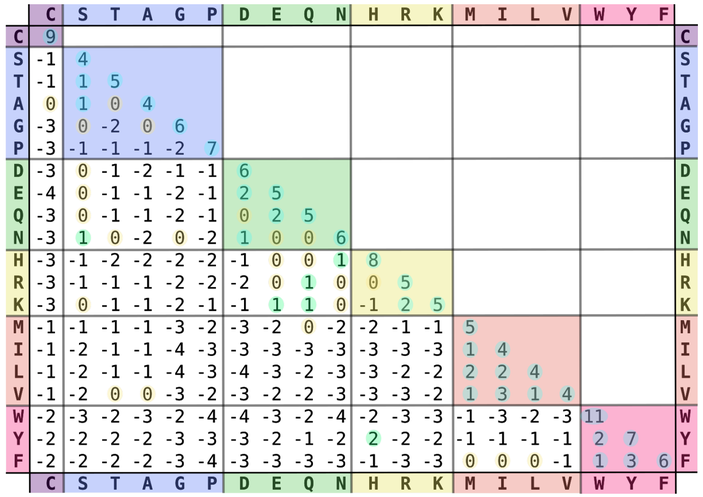
- Match and Mismatch from BLOSUM62
- Gap Opening Penalty: -11
- Gap Extension Penalty: -1
- Algorithm: Needleman–Wunsch

Pairwise Sequence Alignment

P1:	M	A	A	Q	P
P2:	A	A	V	P	

P1:	M	A	A	Q	P
	5	4	0	-11	7
P2:	M	A	V	-	P

Score= 5



	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	

- Match and Mismatch from BLOSUM62
- Gap Opening Penalty: -11
- Gap Extension Penalty: -1
- Algorithm: Needleman–Wunsch

Evolutionary Subword Tokenization

Recall BPE

BPE counts consecutive subwords (pairs) over the entire corpus, finds the most frequent consecutive subwords, merges all instances of the pair and adds the merged subword and merging rule to the vocabulary. BPE repeats this procedure until a target vocabulary size is reached.

Example:

1. **Dataset:** (‘CYN’, 10), (‘FYN‘, 5), (‘FYY’, 12), (‘RYYN’, 4) - **Vocab:** (‘C’, ‘Y’, ’N’, ‘F’, ‘R’)

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

3. **Frequencies: YN → 19**, FY → 17, YY → 16, CY → 10, RY → 4

4. **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’}

Evolutionary Subword Tokenization

Method 1 - Choice

1. Merge the most frequent pair.
2. Choose C candidate pairs from all possible pairs using pair frequencies as their choosability probabilities. (Name comes from numpy.random.choice())
3. Merge maximum of K pairs that align with the most frequent pair above a certain alignment threshold.

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

1. **Dataset:** (‘CYN’, 10), (‘FYN‘, 5), (‘FYY’, 12), (‘RYYN’, 4) - **Vocab:** (‘C’, ‘Y’, ’N’, ‘F’, ‘R’)
2. **Initialize:** (‘C’ ‘Y’ ‘N’, 10), (‘F’ ‘Y’ ‘N’, 5), (‘F’ ‘Y’ ‘Y’, 12), (‘R’ ‘Y’ ‘Y’ ‘N’, 4)
3. **Frequencies:** YN → 19, FY → 17, YY → 16, CY → 10, RY → 4
    - Merge the most frequent pair - **YN**
4. **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.
5. **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 alignment score greater than 0. **(YN, FY) = 1**.
6. **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.

Evolutionary Subword Tokenization

Method 2 - Frequency

1. Merge the most frequent pair.
2. Choose C most frequent pairs as candidate pairs.
3. Merge maximum of K pairs that align with the most frequent pair above a certain alignment threshold.

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

1. **Dataset:** (‘CYN’, 10), (‘FYN‘, 5), (‘FYY’, 12), (‘RYYN’, 4) - **Vocab:** (‘C’, ‘Y’, ’N’, ‘F’, ‘R’)
2. **Initialize:** (‘C’ ‘Y’ ‘N’, 10), (‘F’ ‘Y’ ‘N’, 5), (‘F’ ‘Y’ ‘Y’, 12), (‘R’ ‘Y’ ‘Y’ ‘N’, 4)
3. **Frequencies:** YN → 19, FY → 17, YY → 16, CY → 10, RY → 4
    - Merge the most frequent pair - **YN**
4. **Freq:** Choose the 2 most frequent pairs as candidates (‘F’, ‘Y’) and (‘Y’, ‘Y’)
5. **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**).
6. **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.

Evolutionary Subword Tokenization

Method 3 - Genetic Algorithm

1. Merge the most frequent pair.
2. Apply point-wise mutation (change each aminoacid) with M% to the most frequent pair C times.
    - Add the new mutated pair as candidate if it appears in the dataset.
3. Merge maximum of K pairs that align with the most frequent pair above a certain alignment threshold.

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

1. **Dataset:** (‘CYN’, 10), (‘FYN‘, 5), (‘FYY’, 12), (‘RYYN’, 4) - **Vocab:** (‘C’, ‘Y’, ’N’, ‘F’, ‘R’)
2. **Initialize:** (‘C’ ‘Y’ ‘N’, 10), (‘F’ ‘Y’ ‘N’, 5), (‘F’ ‘Y’ ‘Y’, 12), (‘R’ ‘Y’ ‘Y’ ‘N’, 4)
3. **Frequencies:** YN → 19, FY → 17, YY → 16, CY → 10, RY → 4
    - Merge the most frequent pair - **YN**
4. **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.
5. **Genetic 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.
6. **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.

Evolutionary Subword Tokenization

Alternative Scoring - Frequency Alignment Scoring

Instead of just using alignment score to choose pair from candidates, we can use both 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.

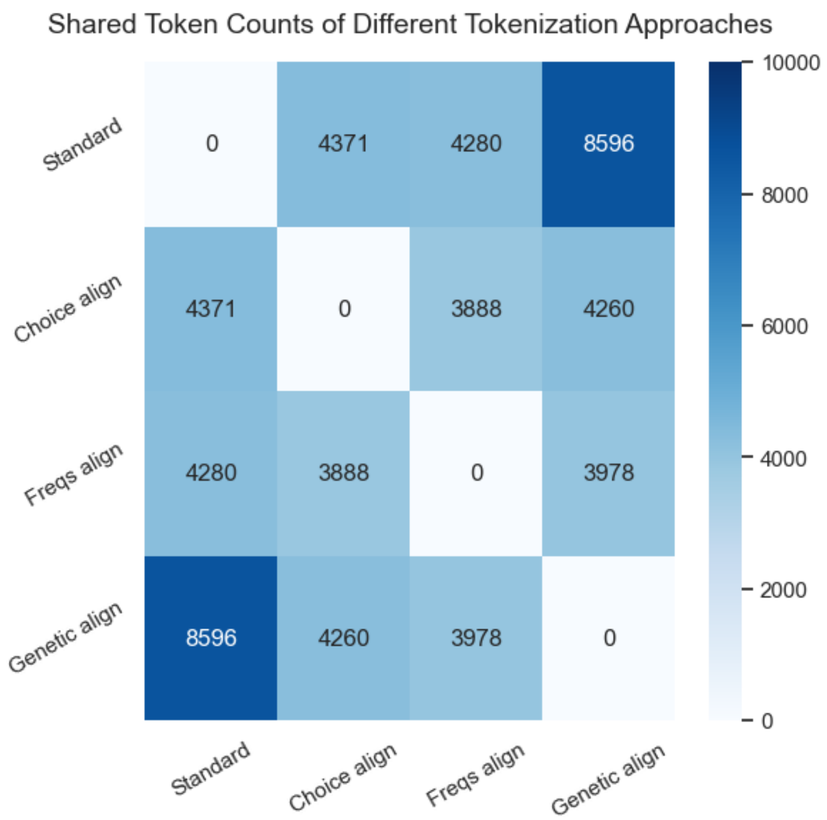
Example (C=2, K=1)**:**

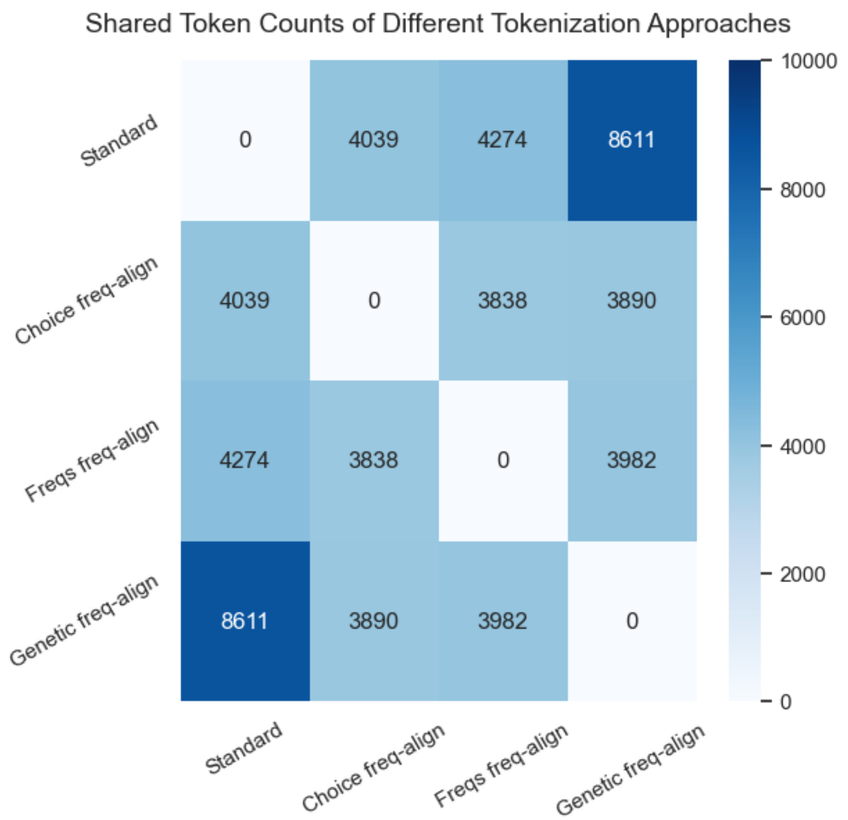
1. **Frequencies:** YN → 19, FY → 17, YY → 16, CY → 10, RY → 4
    - Most frequent pair - **YN**
2. **Freq:** Choose the 2 most frequent pairs as candidates ('F', 'Y') and ('Y', 'Y')
3. **Frequency Alignment Scores (BLOSUM 62, Gap Opening: -11, Gap Extension: -1):**
    - **Alignment Scores:** (YN, FY) = 1, (YN, YY) = 5
    - **Frequency Alignment Scores:** (YN, FY) = (19+17)*1 = 36, (YN, YY) = (19+16)*5 = 175

Experiment Setup

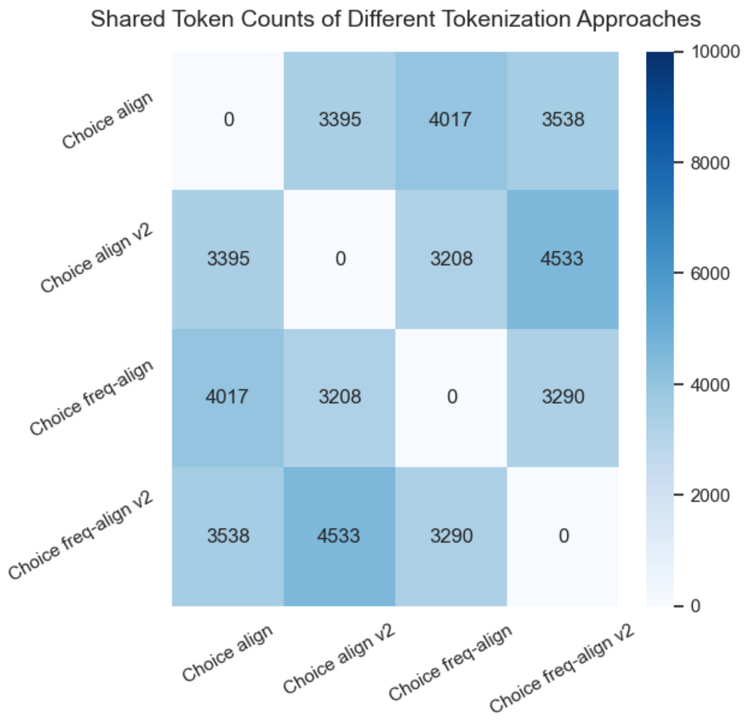
- **Dataset**: Human Taxonomy of UniRef50 - 68978 Protein Sequences
  - The UniRef50 dataset is a subset of UniRef (Universal Protein Resource) and contains representative sequences that are at least 50% identical to any other sequence in the cluster.
- **Vocabulary Size:** 10.000
- **Submatrices:** BLOSUM62 (Main), BLOSUM90 and PAM70
- **Methods:** BLOSUM62, 100 samples, 1 extra merge, Alignment and Frequency Alignment scoring, Method appliance threshold -> 4
- **Alterations:**
  - **Method 1 - Choice:** 2 extra merges, BLOSUM90, PAM70
  - **Method 3 - Genetic:** Mutation percentage 33%

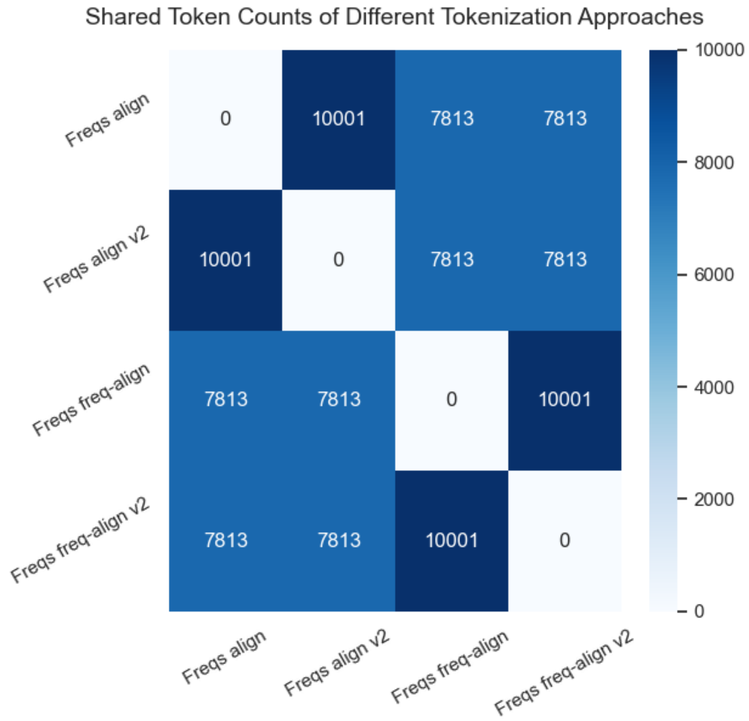
Experiments - Shared Token Counts

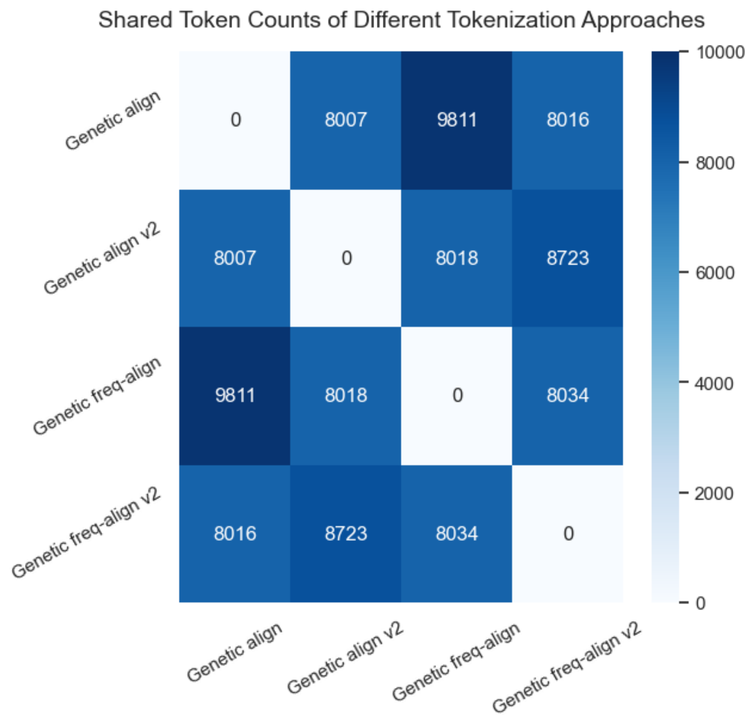




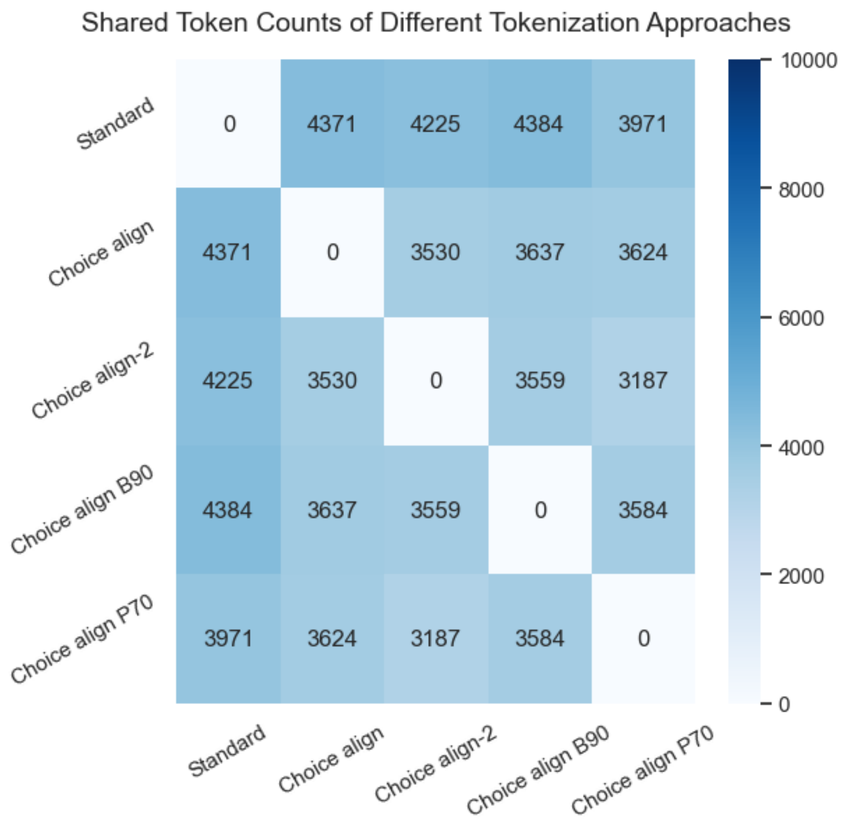
Experiments - Shared Token Counts





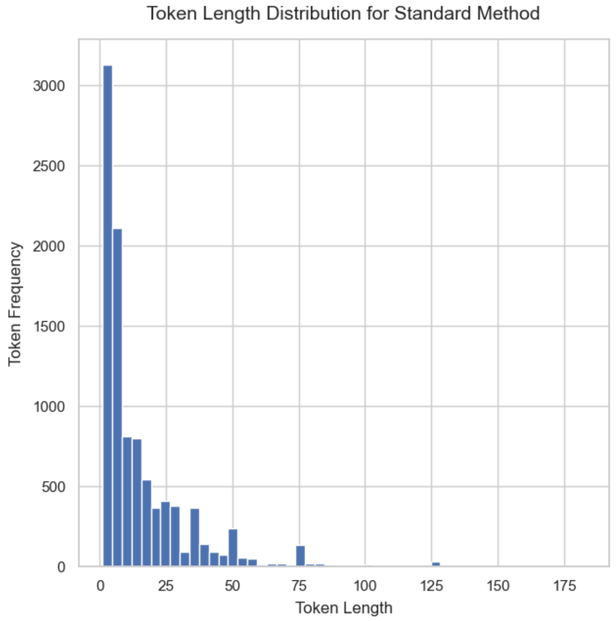


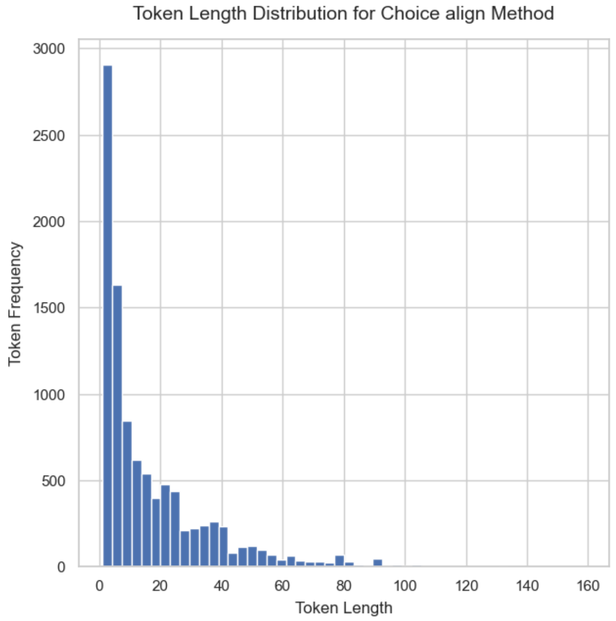
Experiments - Shared Token Counts

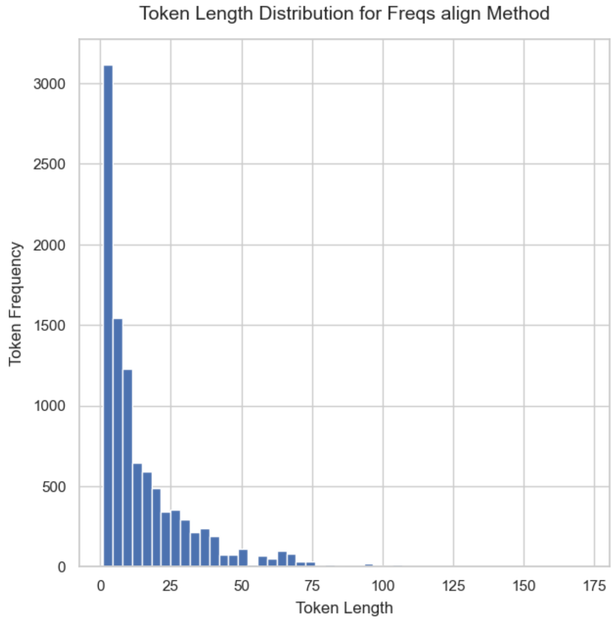


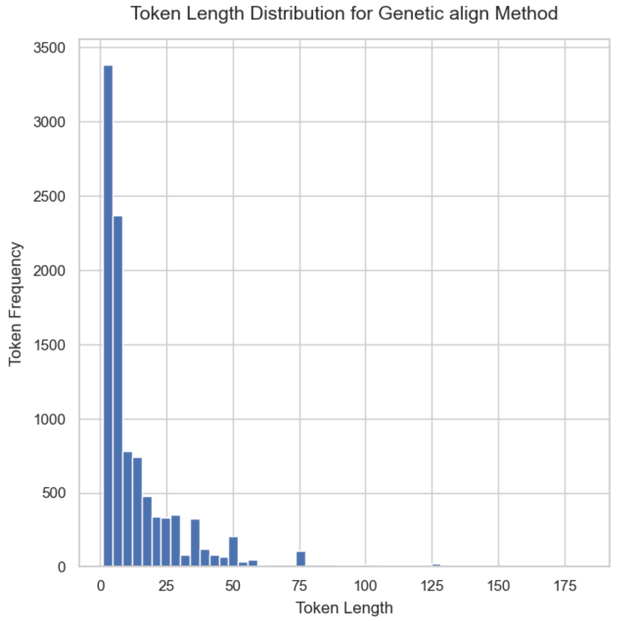
Experiments - Average Token Length

Method	Standard	Choice align	Choice align-2	Choice freq-align	Freqs align	Freqs freq-align	Genetic align	Genetic freq-align	Choice align B90	Choice align P70
Avg Token Length	15.49	17.37	18.49	16.99	15.60	15.58	14.01	14.01	17.53	17.81



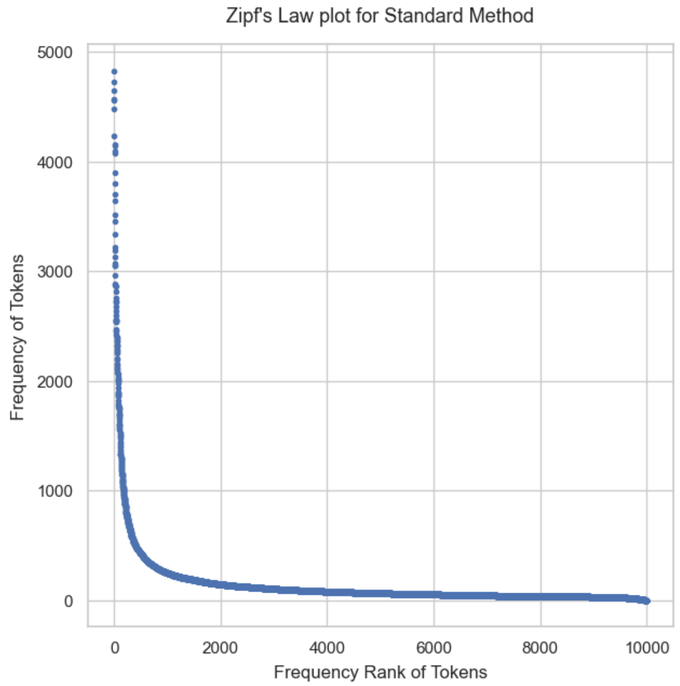


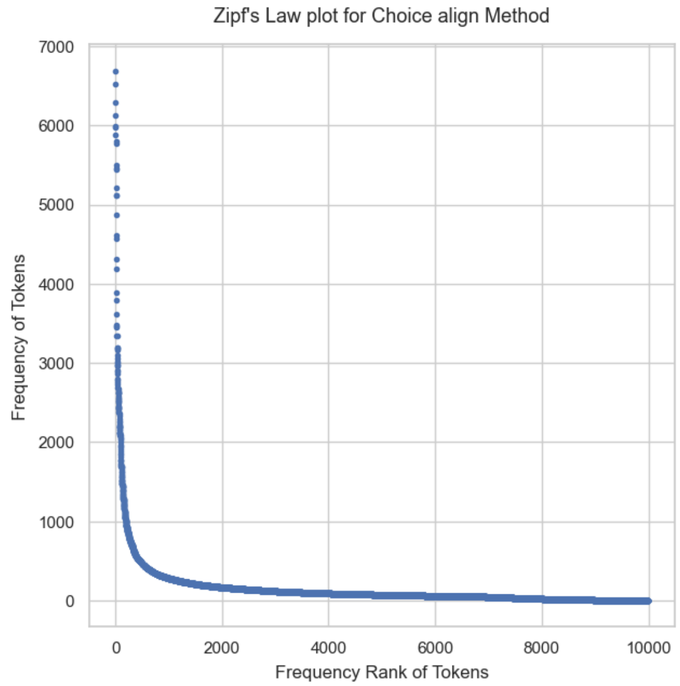




Experiments - Zipf’s Law

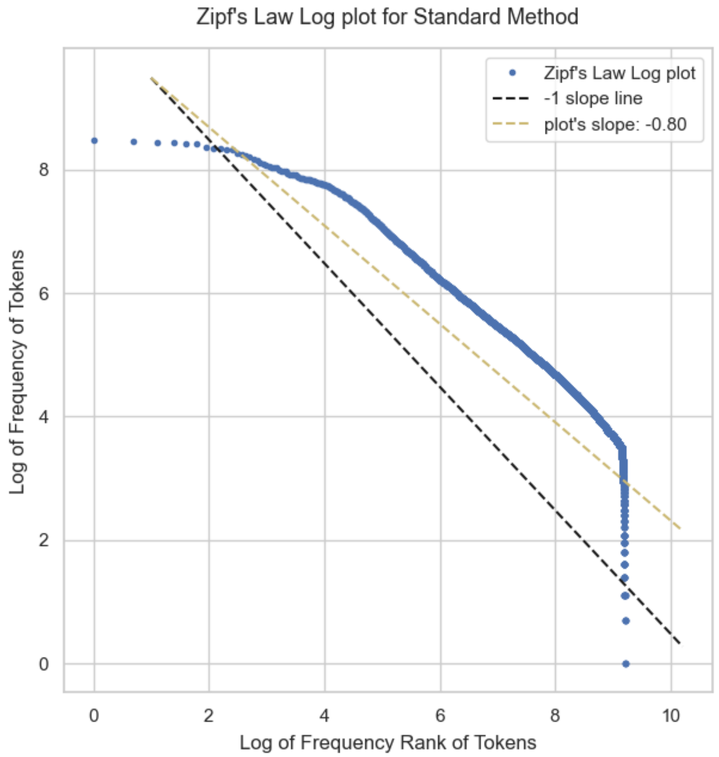
Zipf's Law states that the frequency of a particular element is inversely proportional to its rank. This means that a few elements occur very frequently, while the majority of elements occur infrequently.

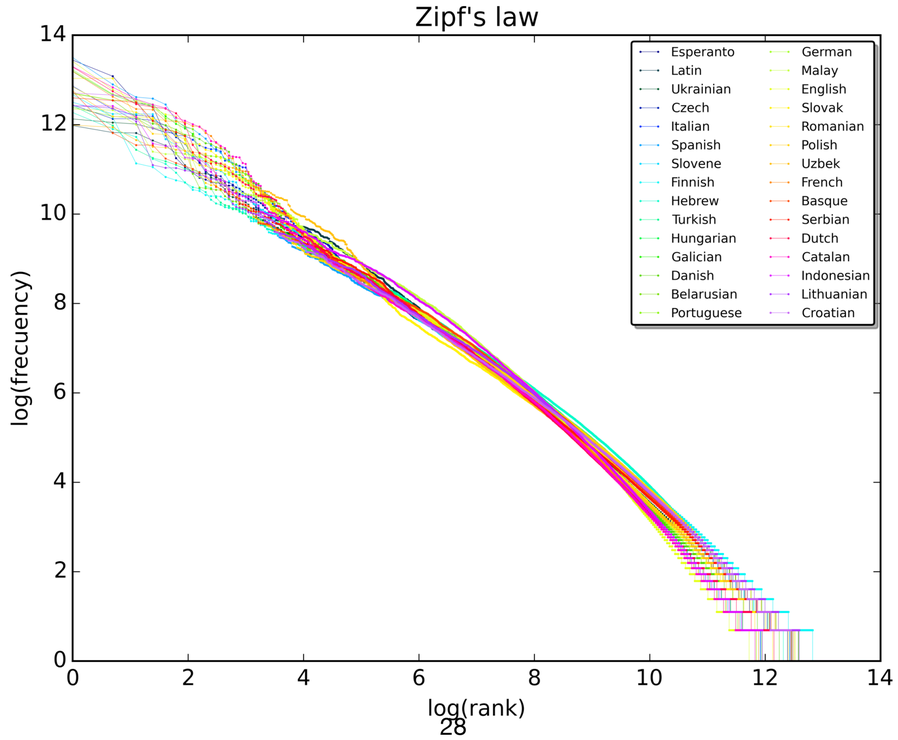


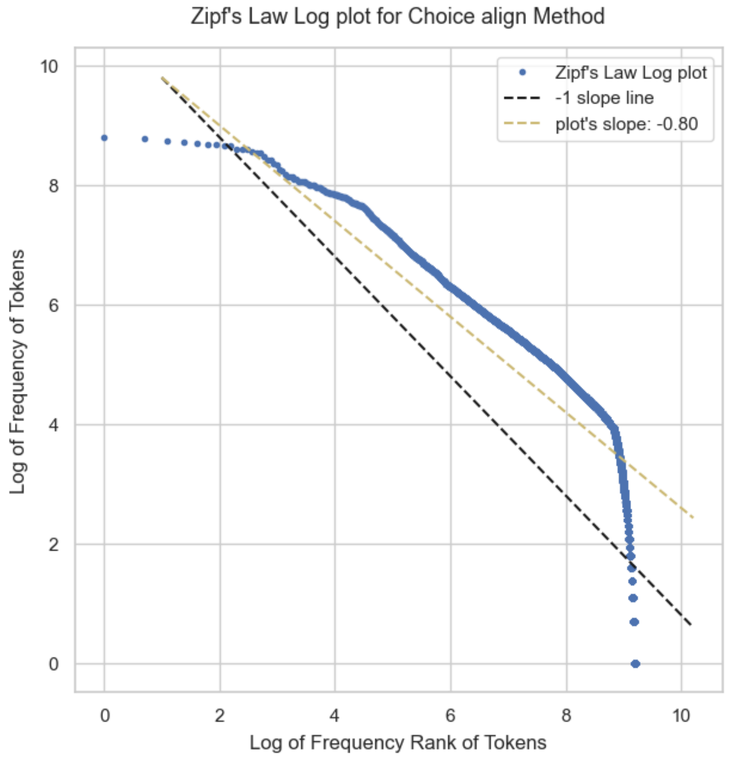


Experiments - Zipf’s Law

Method	Standard	Choice align	Choice align-2	Choice freq-align	Freqs align	Freqs freq-align	Genetic align	Genetic freq-align	Choice align B90	Choice align P70
Zipf’s Slope	-0.80	-0.80	-0.80	-0.79	-0.78	-0.78	-0.80	-0.80	-0.79	-0.79

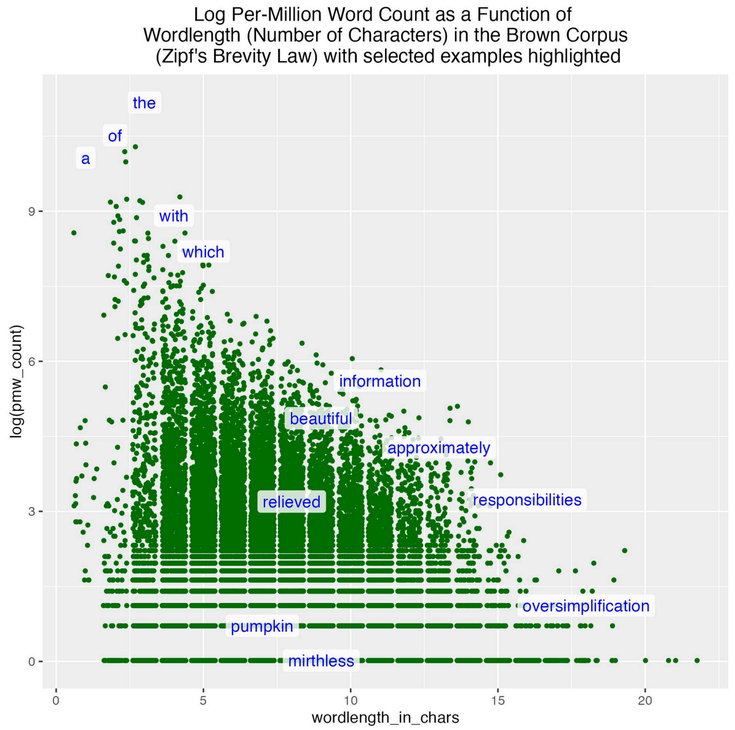




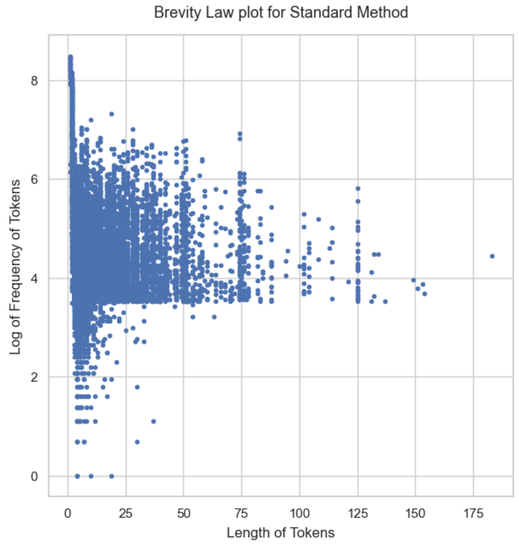


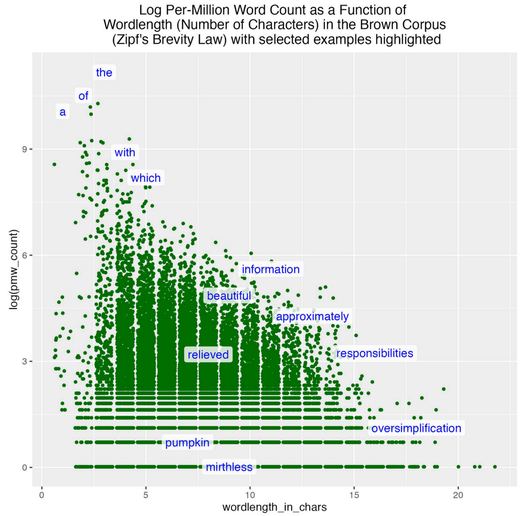
Experiments - Brevity Law (Zipf's law of abbreviation)

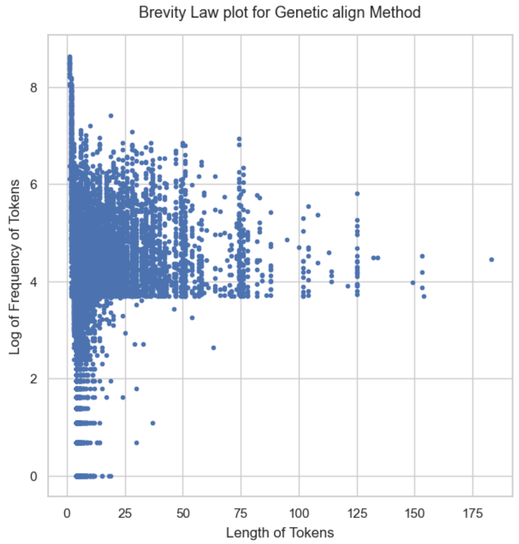
Brevity Law qualitatively states that the more frequently a word is used, the shorter that word tends to be, and vice versa; the less frequently a word is used, the longer it tends to be.

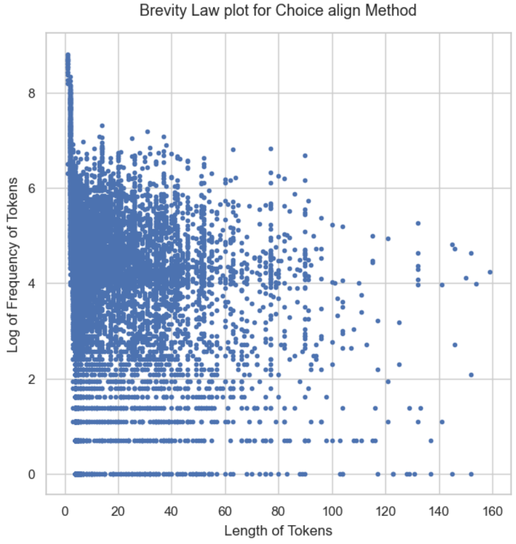


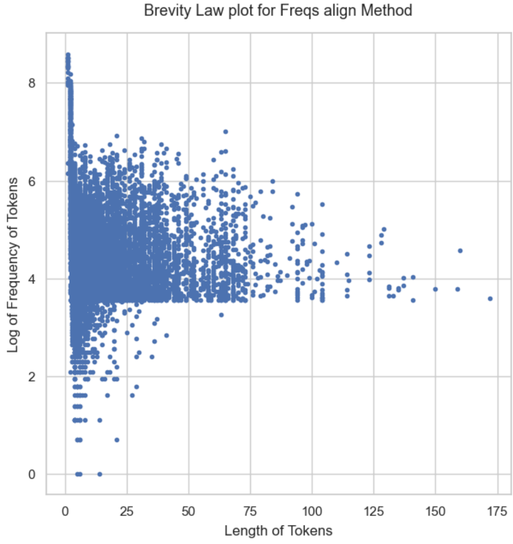
Experiments - Brevity Law (Zipf's law of abbreviation)



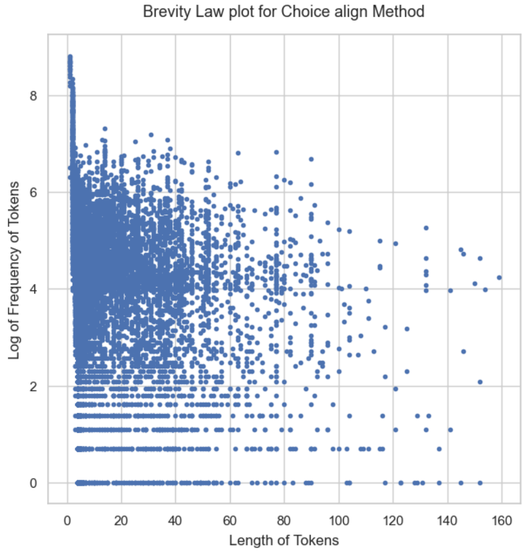


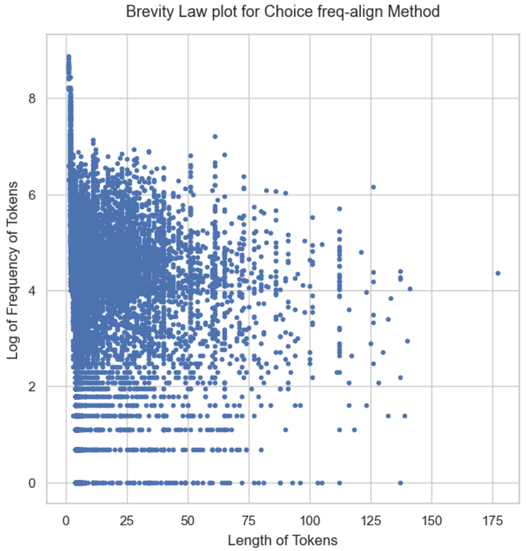


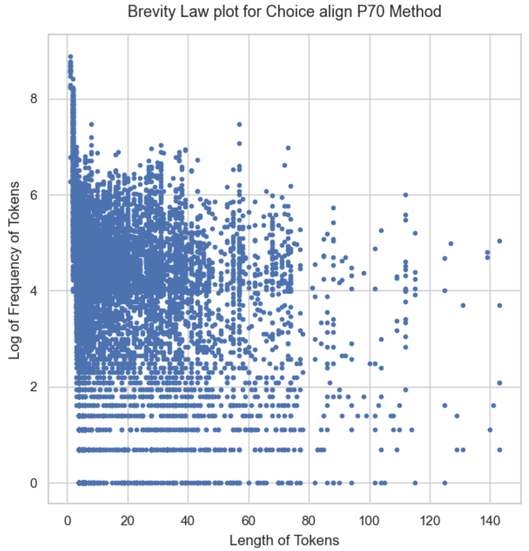


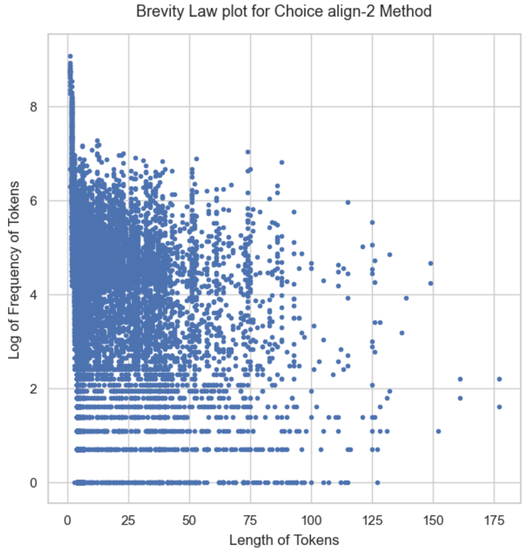


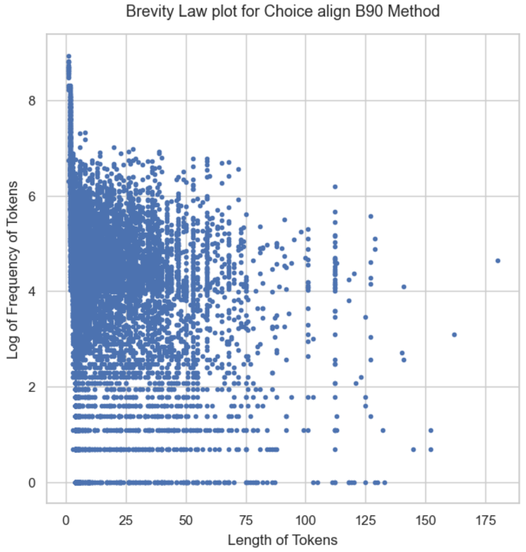
Experiments - Brevity Law (Zipf's law of abbreviation)











Experiments - Heap’s Law (Herdan’s Law)

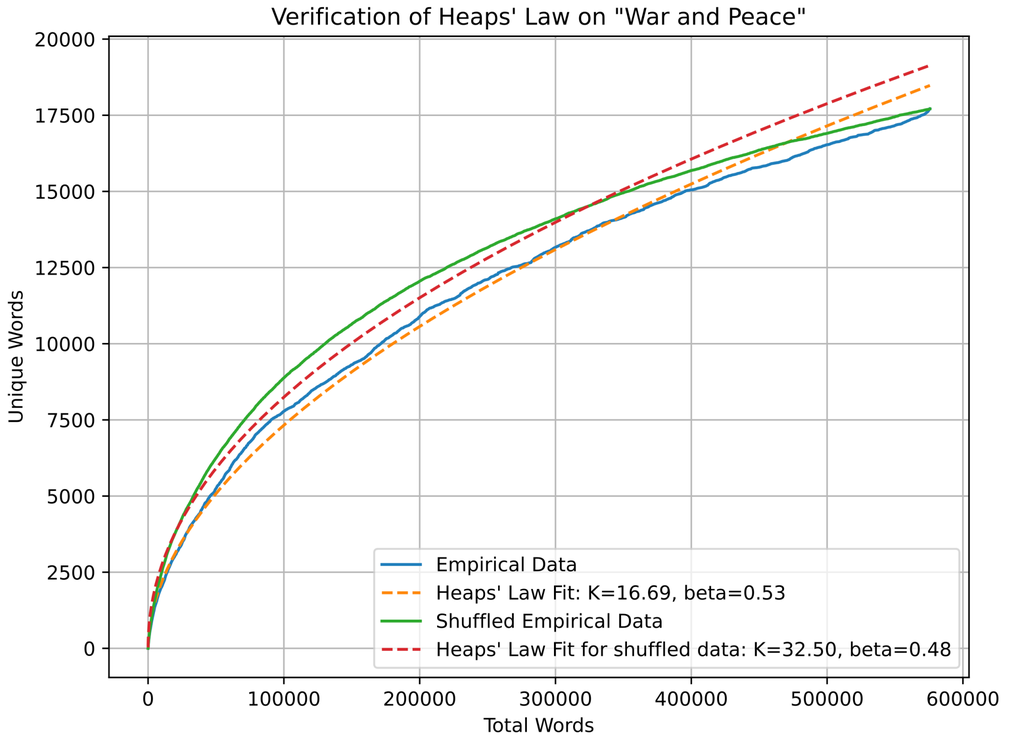
Heap's Law suggests that as the size of a document or dataset increases, the vocabulary size (the number of unique words) also increases, but at a decreasing rate.

The formula is expressed as:

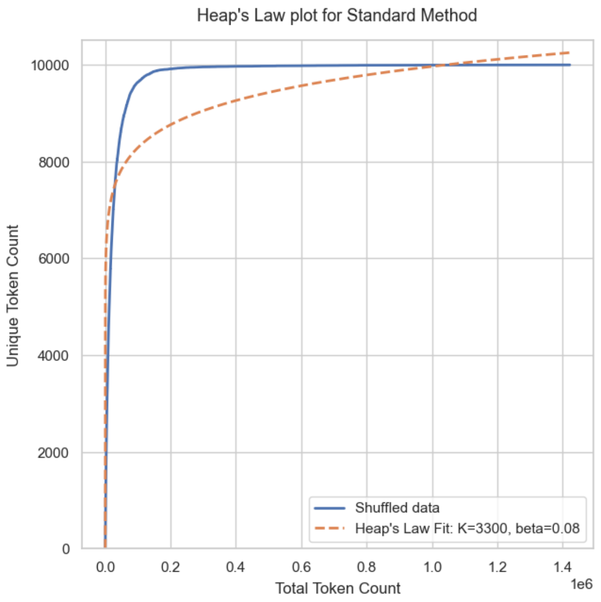
$V(n) = K \cdot n^{\beta}$

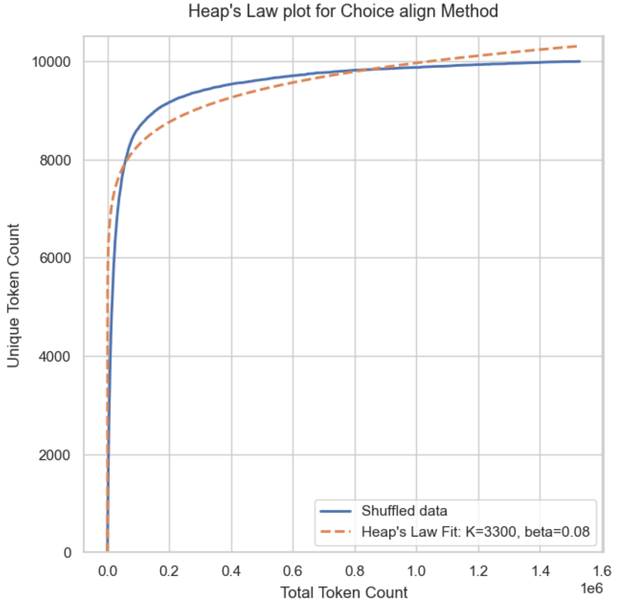
where:

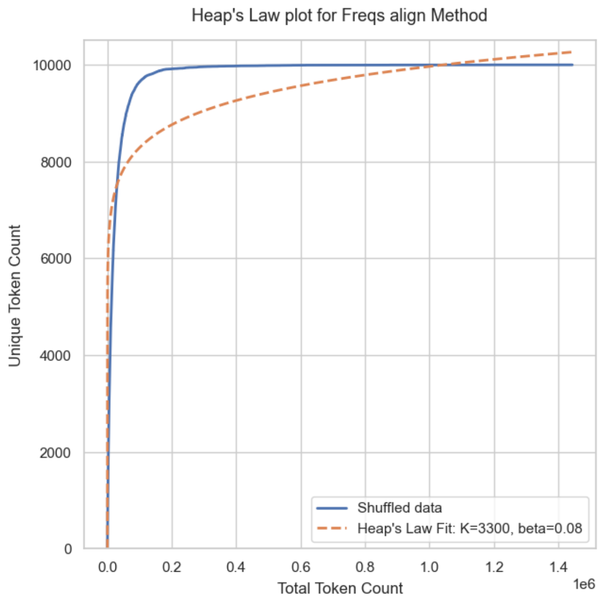
- V(n) is the estimated vocabulary size when the document or collection contains n words,
- K is a constant, typically in the range of 10 to 100.
- β is an exponent, typically in the range of 0.4 to 0.6.

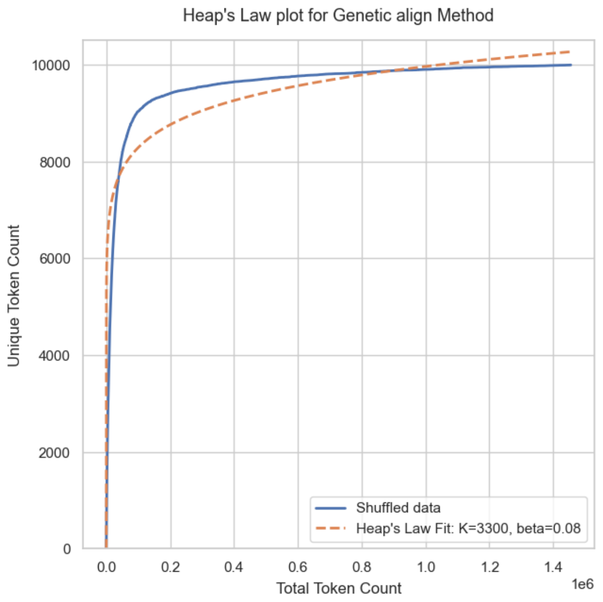


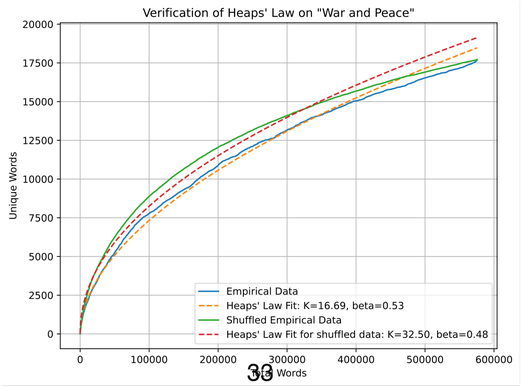
Experiments - Heap’s Law (Herdan’s Law)





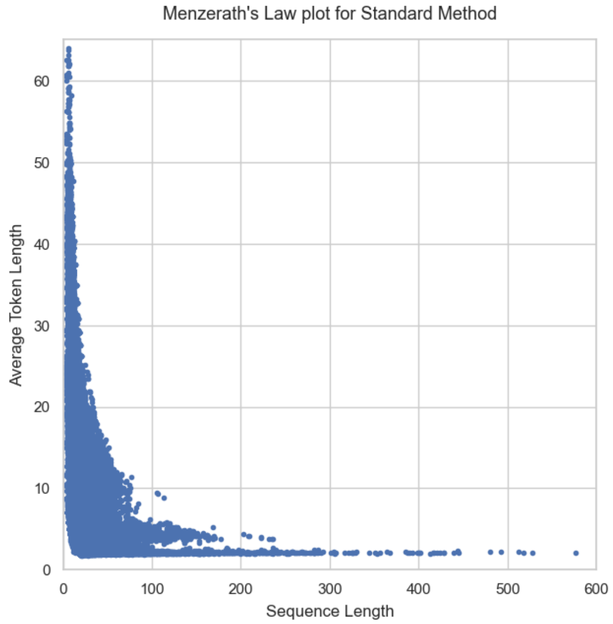


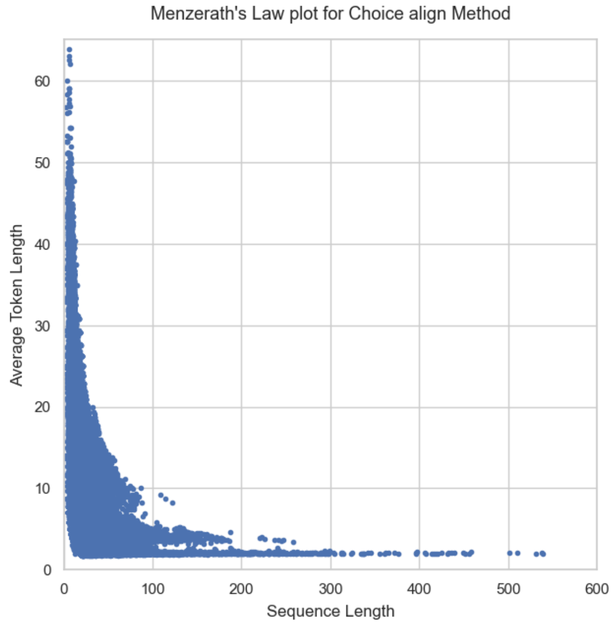




Experiments - Menzerath’s Law (Menzerath–Altmann law)

Menzerath’s Law is an empirical linguistic principle that describes the relationship between the size of linguistic constructs and the size of their constituents. The law is often expressed as follows: "The larger the whole, the smaller the constituents.” For instance, the longer a protein (in tokens), the shorter its tokens (in aminoacids).





Conclusion

- 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.
- **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**.

Future Work

- Optimize the code to run faster.
- Make more experiments, try different hyperparameters.
- Implement the Evolutionary Subword Tokenization approach to WordPiece and Unigram.
- Compare tokens with Protein Domains and Motifs
- Train Protein Language Models with tokens generated from the Evolutionary Subword Tokenization algorithm.
    - Finetune the PLMs for different downstream tasks and compare the results.
- Start working on Graph-based Overlapping and Non-sequential Tokenization.

Thank You For Listening