Natural Langauge Processing Assignment I Report

Elmar Mammadov
Soltan Hasanov

1 Introduction

The project aims to explore the fundamentals of NLP as part of university course work. The idea is to implement various techniques behind the concepts of tokenization, corpus evaluation, sentence segmentation, and spell checking for Azerbaijani texts. The language choice is motivated by the need for more research in the field of language processing in Azerbaijan. The code is written in Python as a single Jupyter Notebook for modular representation of the program.

2 Methodology

The code is composed of separate sections dedicated to tokenization, corpus evaluation, sentence segmentation, byte-pair tokenization, spell checking with Levenstein distance, confusion matrix, and weighted edit distance calculation. The corpus is analyzed with the application of Heaps' law, using the Linear Regression method and Least Square Estimate to find the values of parameters k and β. Using Levenstein distance and BK-tree of $O(log(n))$ search complexity, the spelling check system was implemented. Furthermore, weighted edit distance with weights depending on the values in confusion matrix, based on Azerbaijani keyboard layout, and phonemes analyzed.

3 Experiments

3.1 Whitespace tokenization

Initial experimentation involved processing of the dataset with whitespace tokenization technique. To preserve only meaningful units of data, most punctuation was removed leaving only terminal punctuation, hyphenated words and apostrophes. This tokenization process produced 31457 tokens in total, 11294 of which were unique. Figure 1 shows the most frequent tokens in the corpus. The following regex was used for the separation logic:

$$[r"L + (? : [-''']L+) * "] \tag{1}$$

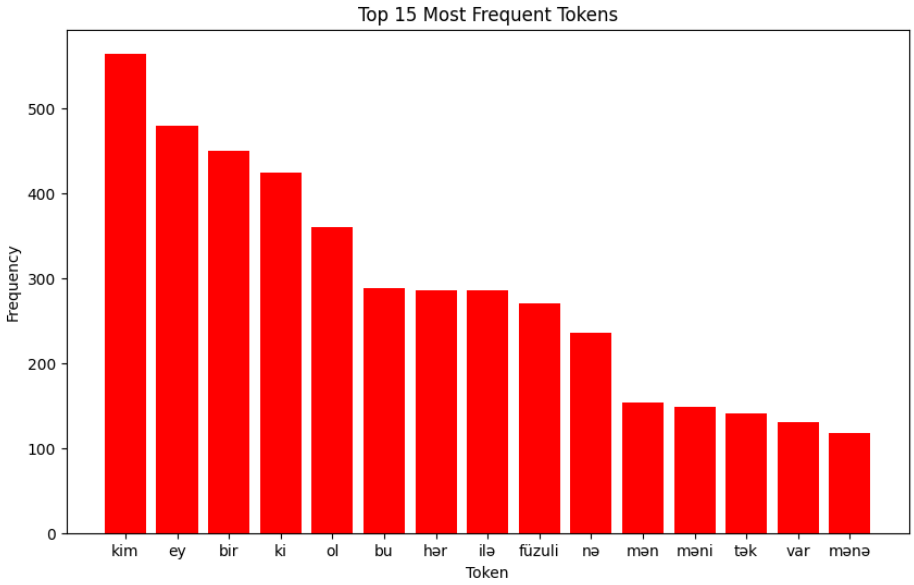

Figure 1: 15 most frequent words

3.2 Heaps' law

The resulting token counts were used to apply the Heaps' Law and visualize the dependence between the corpus and vocabulary sizes. To estimate the values of β and k, the linear regression algorithm is applied on the logarithmic values of V (number of unique tokens) and N (total number of tokens). See the following equations:

$$V(N) = k \cdot N^{\beta} \tag{2}$$

$$\log V = \log k + \beta \cdot \log N \tag{3}$$

$$x_i = \log N_i, \quad y_i = \log V_i \tag{4}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{5}$$

$$\beta = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{6}$$

$$a = \bar{y} - \beta \cdot \bar{x} \tag{7}$$

$$k = e^{a} \tag{8}$$

These calculations produce $k = 2.55$ and $\beta = 0.81$, rounded to two decimal points. Figure 2 shows the growth of the vocabulary size as the total number of tokens in the corpus increases.

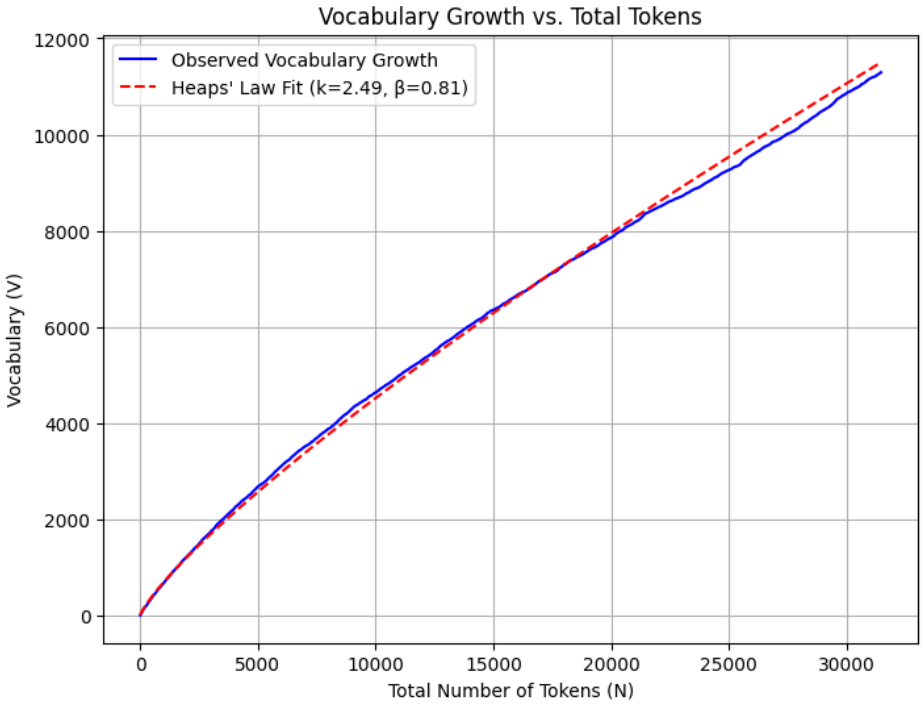

Figure 2: Vocabulary growth curve

3.3 Byte-Pair encoding

The Byte-Pair Encoding (BPE) algorithm was run until 1000 new tokens were added to the dictionary. The longest token produced as a result is "tərcüməsi" with the length of 9 characters. Figure 3 shows the most frequent BPE tokens in the corpus.

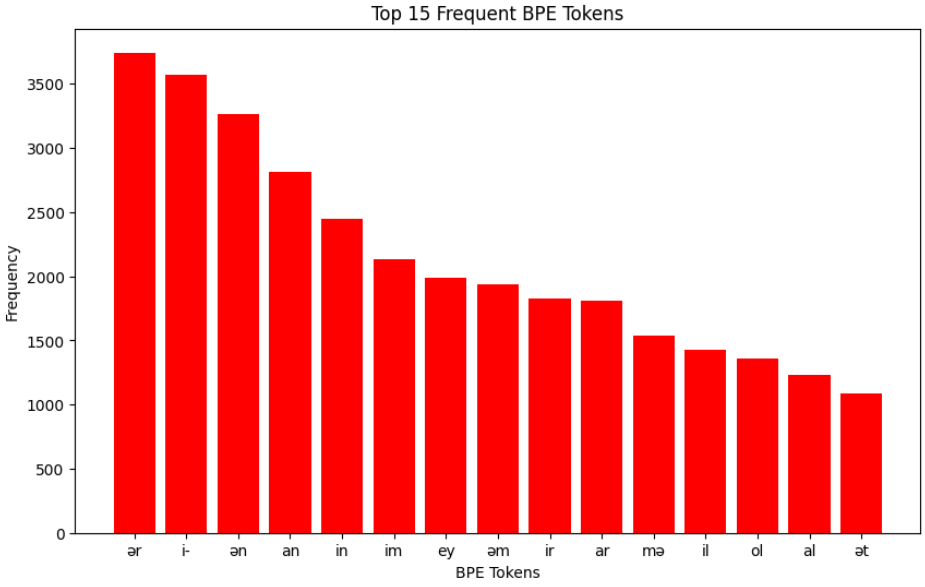

Figure 3: 15 most frequent tokens

3.4 Levenstein distance & Spelling Checker

The spelling check system used Levenstein distance with the whole vocabulary being saved in the BK-tree. The BK-tree provides faster look-up time which is $O(log(n))$ compared to straightforward look-up time $O(n)$. The candidates were ranked based on the Levenstein distance between search word and candidate words. For example, for the search word **lə'li-nabın** the following ranked candidates are provided:

lə'li-nabın	lə'li-nabin	lə'li-nab

While for **ölmaz** we get :

ölməz	olmaz	almaz
olmaq	açmaz	qılmaz
urmaz	olman	olma

3.5 Confusion matrix & Weighted Edit Distance

The second version of the spelling check system uses Weighted Edit Distance. The distance is defined by a formula similar to Levenstein distance, with the substitution value sourced from a confusion matrix and insertion and deletion costs differing based on whether the character is vowel or a consonant. Confusion matrix was build up on the Azerbaijani keyboard layout. The keys were represented in a dictionary data structure, where each letter is represented by (x, y) coordinates and placed at equal distance from its neighbors. Thus, the probability of two letters being confused with each other depends on their distance on a keyboard. From distance, the probabilities are calculated which are the converted to costs using *log* function and its property $log(ab) = log(a) + log(b)$, so that consecutive independent events are represented properly in the weighted edit distance. Additionally, the probabilities of such letters as **ö** and **o** and similar were readjusted as they are often misspelled. The confusion matrix with substituon costs is represented in Figure 4.

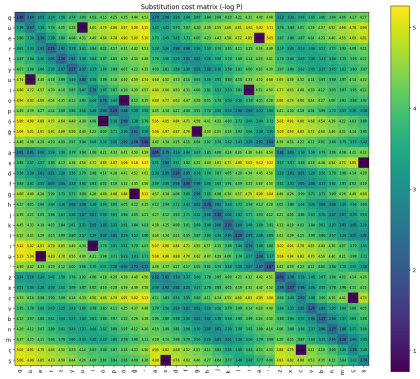

Figure 4: Confusion matrix

For the same examples as before the weighted edit distance based spelling checker suggest the following results.

lə'li-nabın :	lə'li-nabın	lə'li-nabin	lə'li-nab
ölmaz : olmaz	olma	ölməz	almaz

Originally, confusion matrix based spelling check system provided inaccurate results as it didn't consider such misspellings as **ö** with **o**. For this reason, the weights of the probabilities were readjusted several times. In the latest version probabilities of consonants were made higher than that of vowels, and probabilities of **o** being confused with **ö** (and similar cases) were increased. Additionally, special characters were added to the keyboard layout, as the vocabulary included complex words with hyphens and apostrophes. Another problem included different types of apostrophes, which had to be standardized for correct spelling checker candidate suggestions.

4 Contributions

Both members: Tasks 1, 2, 6
Soltan Hasanov: Tasks 3, 4
Elmar Mammadov: Tasks 5, bonus task

5 References

- Jurafsky, D., & Martin, J. H. (2026). Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition with language models (3rd ed., draft). Stanford University & University of Colorado at Boulder.
- Karpathy, A. (2024, February 20). Let's build the GPT tokenizer [Video]. YouTube. https://www.youtube.com/watch?v=zduSFxRajkE
- Grokipedia. (n.d.). BK-tree — Lookup algorithm. Grokipedia. Retrieved from https://grokipedia.com/page/BK-tree#lookup-algorithm
- Füzuli. (2005). əsərləri: I cild [E-book]. Şərq-Qərb. Prezident Kitabxanası. https://www.preslib.az/az/elibrary/ebook/fuEjF361