Azerbaijani Wikipedia Corpus: Tokenization, Heaps' Law, BPE, Sentence Segmentation, and Spell Checking

Team:

Kamal Ahmadov
kahmadov24700@ada.edu.az; kamal.ahmadov@gwu.edu

Rufat Guliyev
rguliyev24988@ada.edu.az; rufat.guliyev@gwu.edu

February 5, 2026

Abstract

This report presents the creation of a large-scale Azerbaijani Wikipedia corpus and an NLP pipeline that includes tokenization, frequency analysis (Zipf's Law), Heaps' law estimation, Byte Pair Encoding (BPE), rule-based sentence segmentation, and a spell-checking system based on Levenshtein distance and weighted edit distance. All processing steps were automated, and the results were saved in structured directories. The report summarizes the methods, experimental setup, results, and key metrics generated during this study.

1 Motivation and Dataset

Goal: The primary objective is to build a comprehensive Azerbaijani corpus from Wikipedia, extract lexical statistics, and create preprocessing modules for common NLP tasks such as tokenization, segmentation, and spelling correction.

Source: The corpus was collected using the MediaWiki API, with cleaning performed through custom scripts. The data was saved as a CSV file at `data/raw/corpus.csv` containing document IDs, titles, revision info, timestamps, URLs, and text.

Corpus Snapshot:

- **Documents:** 31,842
- **Tokens:** 11,905,937
- **Types (unique tokens):** 586,674

Licensing: The corpus and any derived datasets adhere to the CC BY-SA license. All derived artifacts must retain proper attribution.

2 Methods

2.1 Tokenization

We employed a Unicode-aware tokenizer that retains Azerbaijani characters, including internal apostrophes and hyphens, as well as decimal numbers. Non-alphabetic characters are removed, and case normalization (lowercasing) is optional. Additionally, Wikipedia-specific preprocessing removes category/navigation lines and normalizes punctuation (`src/tokenize.py`).

2.2 Frequency Analysis and Zipf's Law

We performed token and type frequency analysis, generating a full frequency table and identifying the top 20 tokens, stored in `outputs/stats/token_freq.csv` and summarized in `summary.json`.

- Zipf's Law was visualized with a rank-vs-frequency plot, available in `outputs/plots/zipf.png`. The plot confirms a typical heavy-tail distribution, which is expected in natural language datasets.

2.3 Heaps' Law

Heaps' Law is used to model vocabulary growth as the corpus size increases. We computed the relationship between the number of tokens N and the number of unique words V, fitting the model $V = kN^{\beta}$ via linear regression in log space (`src/heaps.py`).

- Heaps' parameters: $k = 17.89$, $\beta = 0.640$, indicating moderately fast vocabulary growth.
- The plot of Heaps' law is shown in `outputs/plots/heaps.png`.
- The chosen β value reflects the broad lexical breadth of the corpus, typical of encyclopedic text.

2.4 Byte-Pair Encoding (BPE)

We applied BPE to segment words into subword units, which improves handling of rare and out-of-vocabulary words.

- **Parameters:** 5000 merges, minimum frequency threshold of 2 (`src/task3_bpe.py`).
- Outputs include: `outputs/bpe/merges.txt`, `outputs/bpe/bpe_token_freq.csv`, and a BPE summary in JSON format.
- Results: 6,946 subword types and 430,034 BPE tokens. This reduces the vocabulary size and allows the model to better handle rare words.

2.5 Sentence Segmentation

We implemented an enhanced rule-based approach to sentence segmentation. In this approach, sentence boundaries are detected using punctuation marks and contextual constraints. Specifically, we use the following boundary candidates:

$$p \in \{., !, ?\}$$

We do NOT split when:

$$Non - spaceonbothsidesofp \Rightarrow nosplit$$

$$Examples : \texttt{3.14},\ \texttt{50.5},\ \texttt{S.Rustamov}$$

$$Uppercaseinitial + period \Rightarrow nosplit$$

$$Example : \texttt{A. Mlikli}$$

We split when:

$$Closingquote + space + Uppercaseletter \Rightarrow boundary$$

The enhanced segmentation rules address edge cases, including:

- Abbreviations (e.g., `prof.`, `Dr.`)
- Decimal numbers (e.g., `3.14`, `50.5`)
- Initials (e.g., `S.Rustamov`, `A.M.`)

- Quotes and guillemets: Closing quote followed by space and uppercase triggers sentence boundary
- Lowercase continuation after periods (e.g., `kv. verst`) is preserved to avoid false splits

These heuristics significantly reduce false sentence boundaries, especially in documents with numbers, personal names, and quotes.

Segmentation Evaluation Pipeline

- Tooling: `src/evaluate_segmentation.py` (Precision/Recall/F1 via `sklearn` + BDER), helper wrapper `scripts/eval_sentseg.sh`.
- Inputs: gold and predicted boundary indices (JSON array or newline-separated). BDER is defined as $(FP + FN)/|gold|$.
- Command example: `python -m src.evaluate_segmentation --gold <gold_idx> --pred <pred_idx> --out outputs/sentseg_eval.json`.
- If gold sentences exist (one per line), run `bash scripts/eval_sentseg.sh data/processed/sent_gold.txt 50` to extract subset, segment, convert to indices, and score.
- Status: pipeline ready; manually curated gold boundaries still needed (auto-generated `sent_gold.txt` is a placeholder).
- Reported metrics (once gold is available) are stored in `outputs/sentseg_eval.json`; fields include Precision, Recall, F1, and BDER.

2.6 Checking

We developed an advanced spell-checking system that uses a combination of Levenshtein distance and weighted edit distance to improve accuracy.

The code scans the vocabulary and computes Levenshtein (or weighted) distance to the input word, keeping tokens within a max distance. It sorts candidates by distance, then frequency, and returns the top-k suggestions with their counts. Also, length-based pruning is applied to filter out irrelevant candidates based on character length differences.

Moreover, the spelling checker employs a sophisticated candidate generation mechanism to improve accuracy in handling common mistyped letters in Azerbaijani. Before querying the model with an input word, we identify problematic letters in the word that are most likely to be substituted with their Azerbaijani counterparts.

Candidate Generation:

- We identify problematic letters in the input word (e.g., {a, o, u, c, s, ch, sh, e, g}), which are likely to be substituted with their Azerbaijani counterparts.
- A set of candidates is generated by substituting these characters in the input word in different combinations.
- For each candidate, 5 suggestions are retrieved from the model.
- These lists of suggestions are concatenated to form a combined candidate pool.

Candidate Ranking:

- The lists of suggestions for each candidate are concatenated and sorted by the Levenshtein distance to *any* of the generated candidates and their frequency.
- A list of the top 5 best suggestions is created out of the whole concatenated list and outputted as a result.

This approach significantly boosts the precision of the spell checker, improving success rates for words with common typing mistakes. By leveraging both character-level and frequency-based filtering, the model is able to produce more accurate spelling corrections, even for subtle or frequent letter swaps in the Azerbaijani language.

3 Experiments and Results

3.1 Collection and Cleaning

The corpus was gathered using a random/category fetch from MediaWiki, with templates and non-relevant tags stripped using `mwparserfromhell`. Category/file links were removed, and English-heavy lines were filtered using `langid`. Documents shorter than 400 characters were discarded.

- Final corpus size: 31,842 documents, 11,905,937 tokens, 586,674 types.
- English-heavy lines were filtered to ensure the corpus contained mostly Azerbaijani text.

3.2 Corpus Statistics

The token and type statistics were recorded in `outputs/run_summary.txt`:

- **Tokens:** 11,905,937
- **Types:** 586,674
- Top tokens: Refer to `outputs/stats/summary.json` for the top-20 list.

3.3 Heaps' Law

The vocabulary growth parameter estimates are $k = 17.89$, $\beta = 0.640$, reflecting a vocabulary that grows faster than the classic value of 0.5. This is consistent with the corpus's encyclopedic nature. The Heaps' Law plot is shown in Figure 2.

3.4 Zipf's Law

The Zipf plot in Figure 1 shows a typical rank-vs-frequency distribution, with a linear region confirming Zipf's Law behavior. The dataset's vocabulary distribution follows the expected heavy-tail structure found in many natural language corpora.

3.5 Byte-Pair Encoding (BPE)

The BPE model performed 5,000 merges, generating 5,239 subword types. The total number of BPE tokens emitted was 430,034. This reduces the vocabulary size and facilitates more efficient handling of unseen words. Example segmentations are available in `outputs/bpe/bpe_summary.json`.

3.6 Sentence Segmentation

Sentence segmentation was performed using a rule-based approach tailored to the characteristics of Azerbaijani Wikipedia text. From the first 500 documents, the segmenter extracted 11,479 sentences, producing well-formed sentence boundaries in the majority of cases.

A key challenge in this setting is that punctuation alone is an unreliable indicator of sentence boundaries. Azerbaijani Wikipedia articles frequently contain decimal numbers, abbreviations, personal initials, and quoted material, all of which can lead to erroneous splits under naive punctuation-based rules.

To address these issues, we enhanced the baseline segmenter with several language-aware heuristics. First, periods and commas are not treated as sentence boundaries when they are surrounded by non-space characters on both sides. This prevents incorrect segmentation in numerical expressions (e.g., *3.14*) and compact abbreviations. Second, periods following a single uppercase letter are not interpreted as sentence boundaries, which improves handling of initial-based names such as *K. Ahmadov*. Third, we introduce quotation-aware rules for guillemets and quotation marks. When a quoted segment ends with sentence-final punctuation, the segmenter inspects the following context: if the closing quote is followed by a space and a

lowercase letter, no sentence boundary is introduced; if followed by a space and an uppercase letter, a sentence boundary is detected.

These enhancements substantially reduce false positives while preserving true sentence boundaries, particularly in texts containing numerical expressions, personal names, and embedded citations. Qualitative inspection shows that the enhanced segmenter handles lowercase continuations after periods (e.g., *kv. verst*) correctly, with minimal remaining errors.

3.7 Spell Checking

The Levenshtein-based spell checker, enhanced with weighted edit distance, was evaluated on a synthetic test set of 1,000 misspelled words and achieved an accuracy of Accuracy@1 = 0.489 and Accuracy@5 = 0.726. A confusion matrix of top substitutions is visualized in Figure 3, highlighting common character substitutions.

4 Reproducibility

The full pipeline can be run with the following command:

```
bash scripts/run_all.sh
```

Key outputs:

- Plots: `outputs/plots/zipf.png`, `outputs/plots/heaps.png`
- Stats: `outputs/stats/summary.json`, `outputs/stats/heaps_params.json`
- BPE: `outputs/bpe/merges.txt`, `outputs/bpe/bpe_summary.json`
- Vocab: `data/processed/vocab.txt`
- Spellcheck eval: `outputs/spellcheck/spell_eval.json`, `sample_predictions.csv`, `confusion_heatma`
- Run summary: `outputs/run_summary.txt`

5 Discussion and Future Work

- Enhance data cleaning (address punctuation and diacritics).
- Implement more robust language-ID filtering to exclude non-Azerbaijani content.
- Train a neural segmenter or language model for improved performance.
- Expand spell checker with context-aware methods.
- Replace synthetic evaluation with human-annotated datasets for better performance.

6 Figures

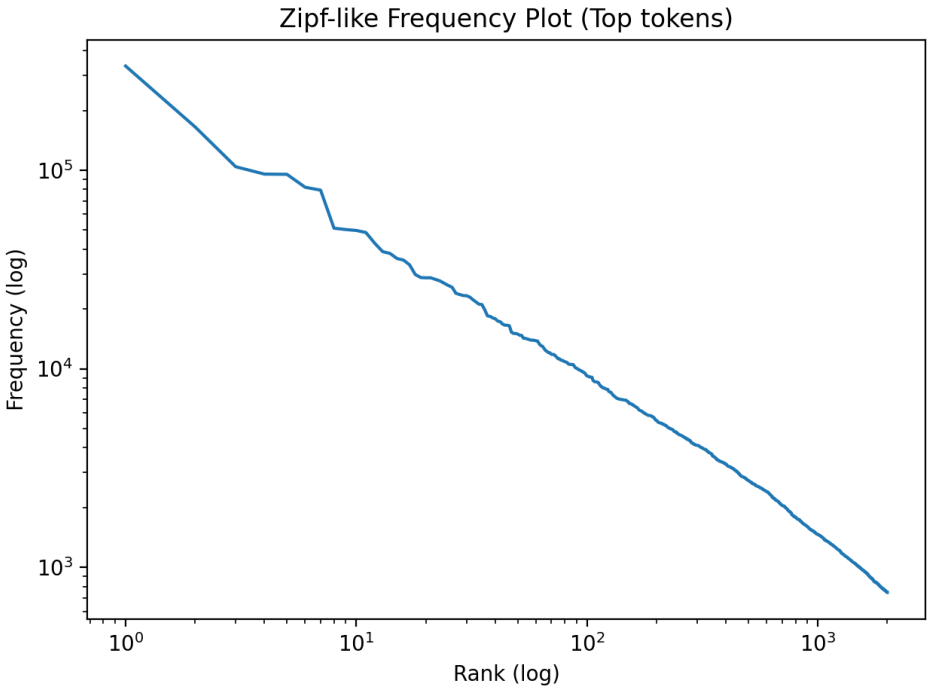

Figure 1: Zipf plot (rank vs. frequency, log–log).

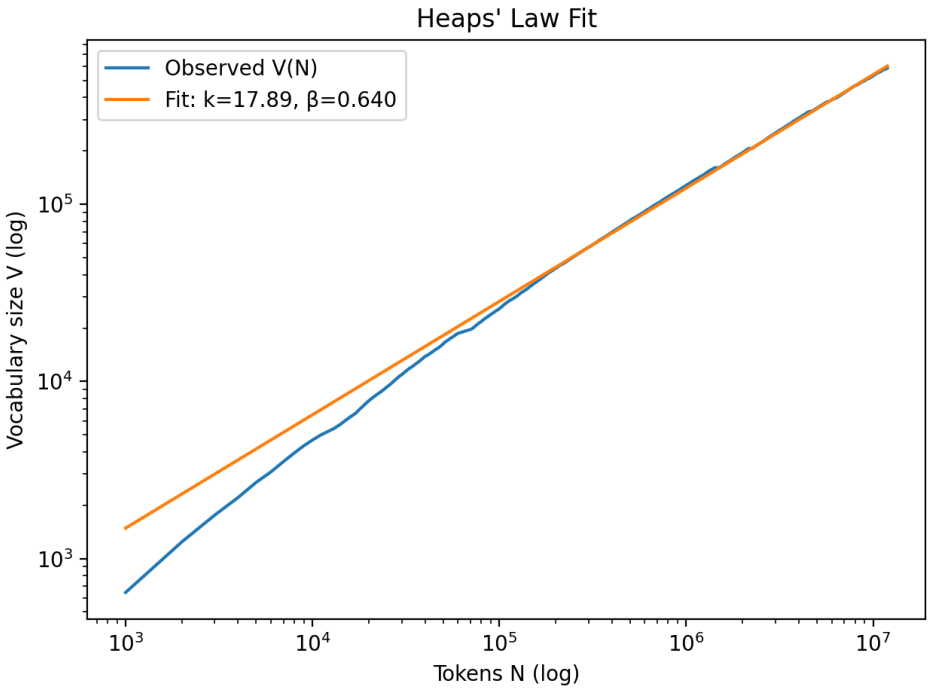

Figure 2: Heaps' law fit with observed $V(N)$ and model kN^{β}.

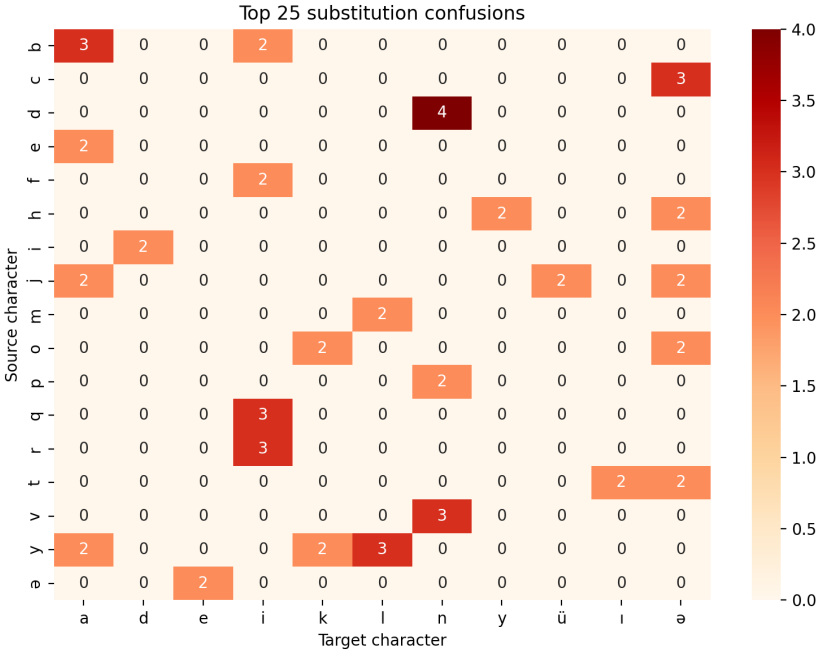

Figure 3: Top substitution confusions (weighted spell checker).