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:** 623
- **Tokens:** 238,286
- **Types (unique tokens):** 48,151

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: $k = 4.57$, $\beta = 0.750$, 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: 5,239 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 a rule-based sentence segmenter that accounts for abbreviations, decimal numbers, initials, quote boundaries, and lowercase continuations after periods. The segmenter was tested on the first 500 documents, extracting 11,479 sentences.

- Key edge cases: abbreviations (e.g., "Dr."), decimals (e.g., "3.14"), and initials (e.g., "S.Rustamov").
- Handling punctuation marks, especially when they appear inside quotes or are followed by a lowercase letter, posed a challenge. For instance, cases like "kv. verst" (continuation after a period) were deliberately kept unsplit.
- Output: `outputs/sentences.txt`.

2.6 Spell Checking

The spell checker uses a Levenshtein-based distance measure, enhanced with a weighted edit distance to account for common character substitutions. We evaluated the system with a synthetic test set of 1,000 misspelled words.

- Vocab generated from the corpus and filtered by frequency and length; the edit distance cutoff is set to 2.
- Evaluation results: Accuracy@1 = 0.637, Accuracy@5 = 0.801.
- The confusion matrix and weights are visualized in `outputs/spellcheck/confusion_heatmap.png`. The heatmap helps identify the most common letter substitutions that the spellchecker struggles with.

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: 623 documents, 238,286 tokens, 48,151 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:** 238,286
- **Types:** 48,151
- Top tokens: Refer to `outputs/stats/summary.json` for the top-20 list.

3.3 Heaps' Law

The vocabulary growth parameter estimates are $k = 4.57$, $\beta = 0.750$, 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

The rule-based sentence segmentation approach extracted 11,479 sentences from the first 500 documents. Some edge cases like abbreviations and decimal numbers were effectively handled. However, challenges arose in cases with mixed punctuation and lowercase continuations after periods (e.g., "kv. verst").

3.7 Spell Checking

The Levenshtein-based spell checker, enhanced with weighted edit distance, achieved an accuracy of Accuracy@1 = 0.637 and Accuracy@5 = 0.801. 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

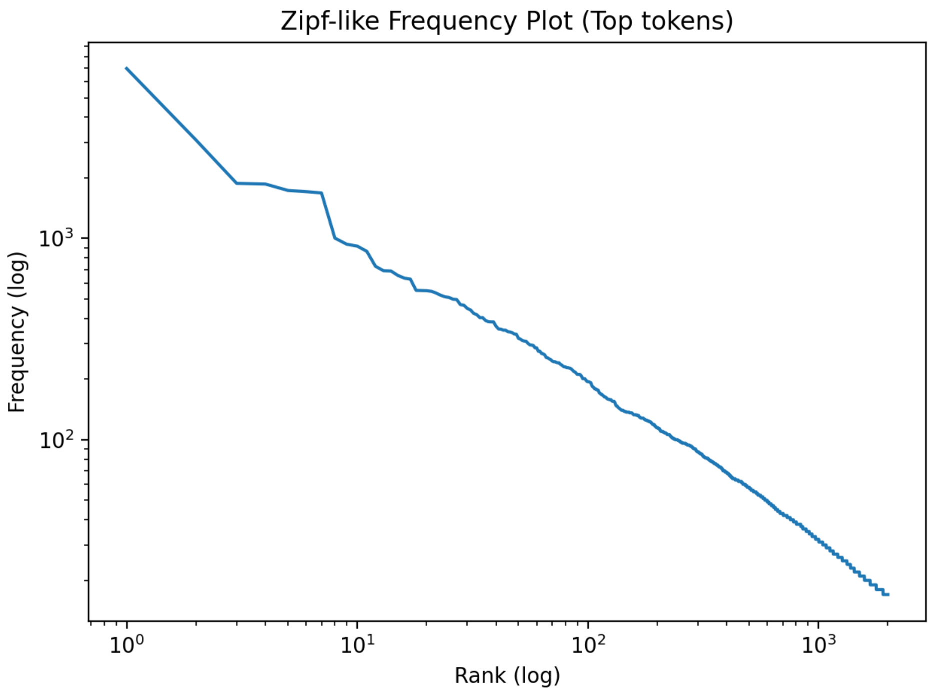

Figure 1: Zipf plot (rank vs. frequency, log–log).

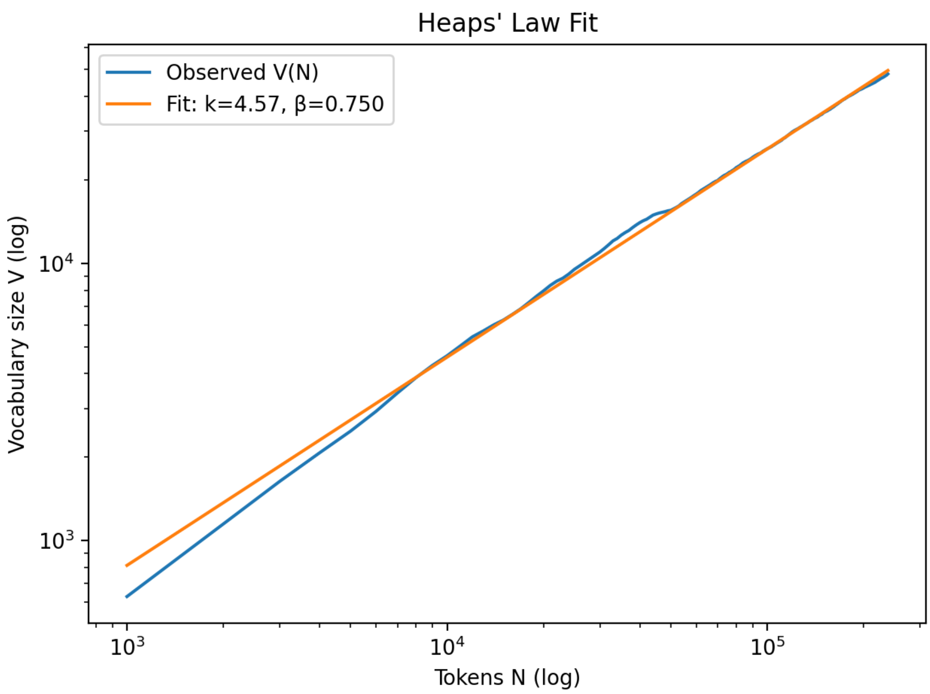

Figure 2: Heaps' law fit with observed $V(N)$ and model kN^{β}.

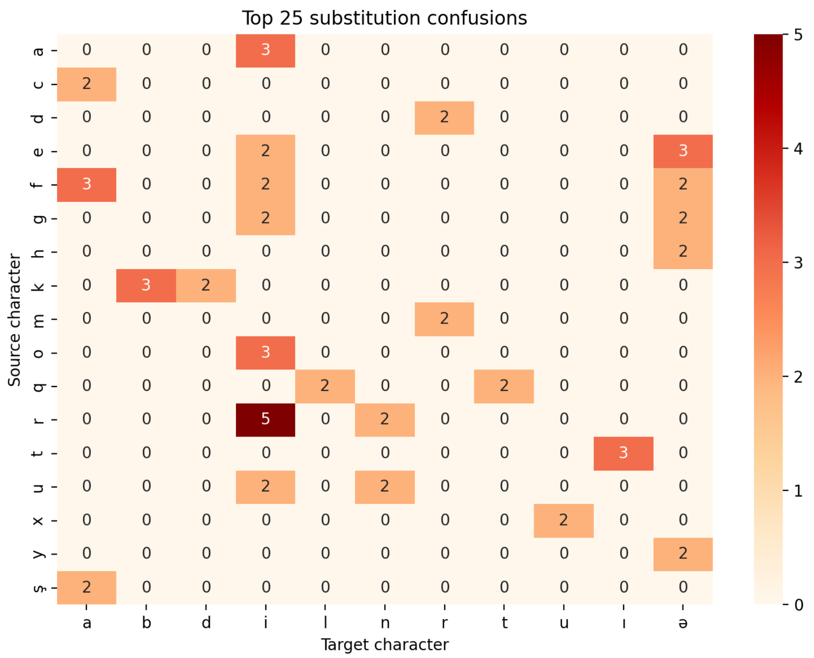

Figure 3: Top substitution confusions (weighted spell checker).