Retrieval-augmented Generation for Academic Research

Milestone 1 Report

Group: QueryMinds
Students: Mohammadali Dehghani, Amir Saadati, Amina Kadic, Meliha Kasapovic

October 28, 2025

1 Introduction

The aim of this paper is to present the process of transforming raw academic PDF documents into a linguistically processed corpus suitable for advanced natural language processing (NLP). As part of the first phase of the project, an automated process was developed that enables text extraction from scientific articles, its cleaning, tokenization, lemmatization and conversion into the standardized CoNLL-U format.

2 Methods

2.1 Text Extraction

We implemented two complementary extraction pipelines:

- **Primary (used in our results): GROBID-based extraction.**
  PDFs were converted to TEI XML using `GROBID`. We then parsed the TEI files to extract core scientific sections (*Title, Abstract, Introduction, and body*) with the `extract_sections.py` script. The output was saved as structured JSON and subsequently sentence-tokenized.
- **Baseline: pdfminer-based extraction.**
  For comparison, `parse_pdfs.py` uses `pdfminer.six` to extract raw text and applies heuristic regular expressions to identify section boundaries. While easy to run, this approach is less robust than GROBID, so it was not used for the final corpus.

We selected the GROBID pipeline for Milestone 1 because it preserves document structure, yields cleaner metadata, and significantly improves section segmentation quality.

2.2 Segmentation and Tokenization

Sentence segmentation and tokenization were performed using the NLTK library in the script `sent_tok.py`. Each section of the document was processed individually using the function `nltk.sent_tokenize()` for sentence splitting and `nltk.word_tokenize()` for word tokenization. The resulting data structure contains each sentence along with its token list and is stored in JSON format in the directory `data/parsed_tokens/`. These files form the basis for the subsequent normalization and lemmatization process. The tokenizer demonstrated high accuracy on most scientific texts, although occasional sentence boundary errors were observed in sections with mathematical expressions or references due to non-standard PDF formatting.

2.3 Normalization and lemmatization

The normalization process included cleaning punctuation, converting text to lowercase, and removing excess spaces. These steps ensured data consistency and reduced noise caused by different formatting of PDF documents. After that, lemmatization and POS-tagging were performed using the `stanza` library, where the `tokenize, pos, lemma` processors were applied. In this way, each word in the corpus received its basic form and grammatical tag, which enabled further linguistic analysis and searching by meaning, and not only by the surface form of the word.

2.4 Conversion to CoNLL-U format

After lemmatization, each processed document was converted to the standardized CoNLL-U format using the `CoNLL.write_doc2conll()` function from the `stanza.utils.conll` library. This format represents a structure where each sentence is written with ten columns containing the token identifier, the original word, the lemma, grammatical tags and dependency relations. All output files are placed in the `data/conllu/` directory, and their correctness was validated using the `conllu` library, which confirmed the structural consistency of the data.

2.5 Quality control

Twenty CoNLL-U files were randomly inspected for accuracy and consistency. The inspection showed that all files were correctly structured and that lemmas and POS tags mostly followed the expected forms. All files were correctly structured, and most tokens had corresponding lemmas and POS tags in accordance with the expected language patterns. Minor differences were observed for domain-specific terminology, foreign words, and complex mathematical expressions, which were sometimes tokenized as multiple separate tokens or partially mislabeled. For example, symbols and expressions such as $\in$, η, or `l(•|u)` are sometimes split into multiple elements, leading to minimal discrepancies in the number of tokens. These phenomena do not significantly affect the quality of the corpus, but indicate opportunities for further segmentation improvements in subsequent phases of the project. The results of the check are documented in the file `docs/qc_notes.md`, while all irregularities and exceptions are recorded in `docs/error_report.md`.

2.6 Corpus statistics

The corpus statistical analysis was performed using the script `corpus_stats.py`, which uses the `pandas` and `conllu` libraries. Basic metrics - number of sentences, total number of tokens, number of unique lemmas and average sentence length - were calculated for each document. The results are stored in the file `docs/corpus_stats.csv`.

3 Results

The processing successfully generated a total of 191 CoNLL-U documents, containing annotated sentences, tokens, and lemmas. The corpus analysis shows that the results are in line with the expected values for scientific texts.

Statistics	Average	Minimum	Maximum
Sentences per document	26	1	107
Tokens per document	853	1	5507
Unique lemmas	665	1	3865
Tokens per sentence	34	1	82

Table 1: Basic corpus statistics calculated on 191 CoNLL-U documents.

The table above shows basic statistics per document. The minimum values for the number of sentences, tokens, unique lemmas, and tokens per sentence are 1, which is the result of incomplete files such as technical abstracts or unsuccessfully parsed PDFs.

4 Conclusion

The first phase of the project successfully implemented the entire data processing process - from text extraction from PDF documents to the formation of validated CoNLL-U files. Using automated scripts data quality and consistency was achieved, which set us up for work on Milestone 2.