MiNI-RAG: A QA System for Faculty-related FAQs Project PoC Report for NLP Course, Winter 2024

Mikołaj Gałkowski, Mikołaj Piórczyński, Julia Przybytniowska
{firstname.lastname}.stud@pw.edu.pl
Warsaw University of Technology, Poland

Anna Wróblewska
Warsaw University of Technology, Poland
anna.wroblewska1@pw.edu.pl

Abstract

This proposal along with PoC outlines the methodology and research framework for developing MiNI-RAG, a Retrieval-Augmented Generation (RAG)-based chatbot designed to address faculty-related FAQs in the Polish language. Over the next two months, the system will be refined through extensive exploration of text segmentation techniques, evaluation of embedding and language models tailored for Polish, and the implementation of the RAGAS framework for performance assessment. Key metrics, including factual accuracy, relevance, and contextual precision, will guide iterative improvements. The project also incorporates a practical proof of concept, demonstrating the feasibility of scraping, processing, and embedding faculty-specific content using advanced tools like ChromaDB and state-of-the-art embedding models. Anticipated challenges include navigating language-specific intricacies and optimizing computational resources. This research aims to advance the application of RAG systems for Polish and similar languages, contributing to their broader adoption and effectiveness.

1 Research Objectives of the Project

In this project, we aim to develop a Polish-language chatbot incorporating a Retrieval-Augmented Generation (RAG) component (Lewis et al., 2020), designed to provide answers to frequently asked faculty-related questions from students. The system will leverage data scraped from the Faculty of Mathematics and Information Science website[1] and other university-related PDF documents that may provide additional helpful information. Through this project, we aim to address the challenges of extracting, organizing, and utilizing domain-specific information from unstructured HTML data to enable intelligent and context-aware responses. The main research questions guiding this project are as follows:

Research Questions:

1. What preprocessing and chunking strategies are most effective to enhance retrieval and generation accuracy?

2. Which embedding models yield the best performance for Polish university-related text data in retrieval tasks?

3. Does prompt engineering can improve the performance?

4. How do Polish LLMs such as Bielik (Ociepa et al., 2024), Krakowiak, and Qra or compare to multilingual counterparts, like Meta Llama-3.1-8B (Dubey et al., 2024) in the context of a chatbot with a RAG pipeline?

5. Which embedding models identified as "best" based on the *Massive Text Embedding Benchmark (MTEB)*[2] (Muennighoff et al., 2023) for Polish data yield the most effective results for retrieval tasks in a considered domain?

6. Is it possible to create a question-answering system for Polish data that is reliable and provides satisfactory answers to user queries with current open-source models and state-of-the-art RAG techniques?

Hypotheses:

[1] https://mini.pw.edu.pl/

[2] https://huggingface.co/spaces/mteb/leaderboard

1. The best-performing embedding model for the Polish language, as identified by the MTEB, will improve the retrieval process compared to the best-performing embeddings designed for English-language content.

2. Properly engineered prompts will significantly enhance the response quality of the RAG system.

3. Evaluation using the RAGAS framework (Es et al., 2023) will provide actionable insights into how our system performs, highlighting specific areas where it struggles the most, as well as offering valuable guidance for refining retrieval and generation pipelines.

2 Significance of the Project

State of the Art: Retrieval-Augmented Generation (RAG)-based systems (Lewis et al., 2020) represent the forefront of conversational AI, particularly in domains requiring accurate and efficient information retrieval (Gao et al., 2023). While extensive research has been conducted on English-language models, there is a notable gap in adapting these techniques to Polish-language contexts, especially within academic and institutional domains.

Justification for Tackling the Problem: The Polish-language chatbot ecosystem remains underdeveloped, with existing solutions often relying on generic translations that fail to deliver optimal performance. This gap leaves a significant opportunity to develop tailored systems capable of meeting the specific linguistic and contextual needs of Polish-speaking users.

Students at the Faculty of Mathematics and Information Science frequently encounter challenges and spend considerable time searching for critical information such as the dean's office opening hours, contact details for faculty members, or academic regulations available on the faculty's website. These inefficiencies highlight the need for a streamlined, user-friendly system capable of delivering precise answers to frequently asked questions. By addressing these challenges, *MiNI*-RAG aims to enhance the overall user experience and save valuable time for both students and staff.

Pioneering Nature:

This project is pioneering in its application of Retrieval-Augmented Generation (RAG) to Polish-language academic content, a relatively underexplored area in natural language processing. While RAG-based systems have shown impressive results in English and other widely spoken languages, there is a significant gap in their adaptation to Polish, especially for domain-specific use cases such as answering faculty-related queries. Most existing systems are either language-agnostic or focused on English, and often fail to account for the unique linguistic and cultural aspects of Polish, which can lead to suboptimal performance in local contexts.

By focusing specifically on Polish, this project will provide a unique solution tailored to the language's nuances, bridging the gap between unstructured web data and efficient, context-aware conversational interfaces. Additionally, it will address a pressing need in higher education: a reliable and efficient method for students and faculty to retrieve relevant, often time-sensitive information. The combination of RAG technology with Polish-language data and academic contexts sets this project apart, and its results will be pioneering in both practical and theoretical terms, contributing to the development of domain-specific, language-adaptive AI systems.

Impact: The results of this project will advance:

1. **Evaluation of RAG and LLMs for Polish Data:** Providing a comprehensive assessment of how Retrieval-Augmented Generation (RAG) systems and Large Language Models (LLMs) perform on Polish-language data, contributing to the understanding of their current capabilities and limitations in non-English contexts.

2. **Improving User Experience for Faculty-Related Queries:** Automating the process of finding information in a more user-friendly way (Neupane et al., 2024), addressing the specific needs of students and staff at our faculty, such as retrieving details about the dean's office hours or faculty contact information.

3. **Future Research Directions:** Offering insights and benchmarks that can inform future improvements in NLP techniques for Polish and similar languages, as well as inspire further research into making RAG systems more effective and domain-specific.

3 Concept and Work Plan

General Work Plan:

1. **Initial Proposal and SOTA Analysis (By November 27):** Submitting the project proposal, including a state-of-the-art (SOTA) analysis of literature, tools, open-source pretrained models, and datasets.

2. **Data Collection and Preprocessing (completed on November 26):** Scraping data from the Mini PW website and preprocessing HTML, including cleaning, handling special characters, and parsing content.

3. **Proof of Concept (PoC) and Literature Review (By December 11):** Testing preliminary chunking strategies and embedding models. Trying out some LLMs to create a basic chatbot PoC, which will not include evaluation or optimization at this stage. Writing a literature review on Polish-focused embedding models, RAG systems, and LLMs. Preparing a preliminary report and collecting feedback.

4. **LLM Integration and Refinement (By January 5):** Experimenting with LLMs for Polish, optimizing prompt engineering, and refining chunking strategies based on PoC results.

5. **Evaluation and Optimization (By January 15):** Using the RAGAS framework to evaluate system performance, identifying strengths and weaknesses, and iterating to improve retrieval and generation pipelines.

6. **Final Presentation and Report (By January 22):** Preparing the final project presentation and compiling a detailed report, documenting methodologies, results, and future work directions.

Risk Analysis:

- **Language-Specific Challenges:** Limited Polish-language NLP resources may hinder model performance. *Mitigation:* Focus on pre-trained models optimized for Polish.

- **High Computational Requirements:** Embedding models and LLMs may require significant computational resources, potentially limiting model performance or scalability. *Mitigation:* Explore efficient embeddings and utilize servers with GPUs to scale up computation.

- **Uncertainty of Model Performance on Polish Data:** There is a risk that the system may not perform well on Polish data, potentially failing to retrieve relevant chunks or generate meaningful responses. *Mitigation:* Thoroughly test and evaluate different models and strategies in the early stages, and make necessary adjustments based on feedback and performance results.

4 Approach & Research Methodology

Data Preprocessing: HTML scraping with Python libraries (BeautifulSoup, Selenium). Data cleaning: Removing boilerplate content, normalizing Polish text.

Chunking: In line with common practices in the literature (Gao et al., 2024), we are experimenting with various chunking techniques to split the data into manageable pieces. Typically, documents are divided into fixed-size chunks based on a specific token limit (e.g., 100, 256, 512), as larger chunks capture more context but also introduce noise and increase computational costs. Smaller chunks reduce noise but may fail to convey complete context, often leading to sentence truncation. To address this, advanced techniques like recursive splitting and sliding windows have been proposed to optimize chunking and enhance retrieval performance by merging globally related information across multiple retrieval steps. Inspired by these approaches, we aim to evaluate the following methods:

- **Length-Based Chunking (Character Count):** This method divides text into fixed-size chunks, each containing a predefined number of characters. While computationally efficient, it risks splitting sentences or words, leading to fragmented chunks.

- **Text-Structured-Based Chunking (Recursive Character Splitter):** This approach retains structural integrity by recursively splitting text at logical boundaries (e.g., paragraphs, sentences) until the chunks meet size constraints. It offers a balance between size and context retention.

- **Semantic-Meaning-Based Chunking (Optionally):** We plan to explore NLP techniques, such as embedding-based similarity, to create semantically coherent chunks. This method is computationally intensive but ensures contextual relevance.

Embedding & Retrieval: Retrieval in RAG systems is typically achieved by calculating the similarity (e.g., cosine similarity) between the embeddings of queries and document chunks. The semantic representation capability of embedding models plays a critical role in this process. According to recent surveys (Yu et al., 2024), both sparse and dense retrieval methods have their strengths. Sparse retrieval models like BM25 excel at capturing term-based relevance, while dense retrievers, often based on pre-trained language models (PLMs) such as BERT, provide rich semantic representations. Recent advancements, including multi-task instruct-tuned models like BGE (Xiao et al., 2024) or GTE (Li et al., 2023), have significantly improved performance across various tasks. The MTEB leaderboard evaluates embedding models on a wide array of datasets, highlighting their strengths in specific applications.

For now, we focus on semantic search using dense retrieval methods. However, hybrid approaches that combine sparse and dense retrieval, as suggested in the literature, show promise for enhanced robustness and efficiency. In such systems, sparse retrievers can improve zero-shot performance and handle rare entities, complementing dense models. This approach could be particularly useful for our use case, where keyword-based initial searches could refine dense retrieval results.

Polish-centric Embedding Models:

- **BGE-Multilingual-Gemma2:** A multilingual embedding model based on the Google/Gemma-2-9B LLM. Trained on diverse languages and tasks, it demonstrates advancements in multilingual text embeddings.

- **gte-Qwen2-7B-instruct:** The latest in the gte (General Text Embedding) family, ranking No.5 on the MTEB benchmark for English and No.6 for Chinese as of the end of November 2024. (Yang et al., 2024)

- **gte-Qwen2-1.5B-instruct:** A smaller version of the gte-Qwen2-7B, built on the Qwen2-1.5B LLM, following the same training methodology as its larger counterpart.

- **jina-embeddings-v3:** A multilingual, multitask embedding model based on the Jina-XLM-RoBERTa architecture. It supports long input sequences and task-specific embedding generation using LoRA adapters.

- **MMLW:** A neural text encoder for Polish, based on RoBERTa and trained with multilingual knowledge distillation. It generates embeddings for various NLP tasks, including semantic similarity and clustering.

General-purpose English-centric Embedding Models:

- **NV-Embed-v2:** A generalist embedding model ranking first on the MTEB benchmark, with top scores in retrieval tasks crucial for RAG systems. (Lee et al., 2024)

We are utilizing vector search tools such as Chrome, Milvus, or similar frameworks for retrieval. While our primary focus is on semantic search, we acknowledge the potential benefits of hybrid retrieval strategies and may explore their implementation in future iterations.

Generation: Testing Polish-optimized LLMs and experimenting with prompt templates to enhance response quality. We will also assess how Polish large language models, such as Bielik (Ociepa et al., 2024), Krakowiak (Ruciński, 2023), and Qra [3], perform in the context of a chatbot with a RAG pipeline. Additionally, we will evaluate multilingual LLMs like Meta Llama-3.1-8B (Dubey et al., 2024) for their ability to handle Polish queries and generate relevant, coherent responses within this setup.

Evaluation: Based on insights from the recent survey on Retrieval-Augmented Generation (RAG) evaluation methodologies (Yu et al., 2024), we concluded that the *RAGAS framework* (Es et al., 2023) offers the most comprehensive and suitable approach for our project. This framework provides both quantitative and qualitative metrics to assess precision, recall, relevance, and coherence, which align well with our evaluation needs.

To complement this, we will create an evaluation dataset by gathering the most frequently asked and intriguing questions from students, potentially

[3] https://huggingface.co/OPI-PG/Qra-7b

using a Google Form for input collection. This will allow us to evaluate the chatbot in realistic, user-centered scenarios.

We will assess the performance of our chatbot using metrics commonly used in retrieval-augmented generation literature (Yu et al., 2024) such as:

- **Factual Correctness:** This metric compares and evaluates the factual accuracy of the generated response with a reference. The score ranges from 0 to 1, with higher values indicating better performance. The alignment between the response and the reference is assessed by first breaking down the response and reference into claims, then using natural language inference (NLI) to determine the factual overlap. Precision, recall, and F1 score are used to quantify factual overlap.
- **Context Precision:** This metric measures the proportion of relevant chunks in the retrieved contexts. It is calculated as the mean of precision@k for each chunk in the context. Precision@k is the ratio of relevant chunks at rank k to the total number of chunks at rank k.

$$\text{Context Precision} = \frac{\sum_{k=1}^{K} Prec@k \times v_k}{A}$$

$$Prec@k = \frac{true\ posit@k}{true\ posit@k + false\ posit@k}$$

, where:

  - `A` - Total Number of Relevant Items in Top K results
  - `K` - total number of chunks in `retrieved context`
  - v_k - relevance indicator at rank k

- **Context Recall:** Measures how many of the relevant documents were successfully retrieved. Higher recall means fewer relevant documents were missed. Recall is important to ensure no key information is overlooked. It is calculated using reference data for comparison.

$$\text{Context Recall} = \frac{|\text{GTCTCBATC}|}{|\text{Claims In GT}|}$$

, where `GT` stands for Ground Truth, `GTCTCBATC` stands for GT Claims That Can Be Attributed To Context

- **Noise Sensitivity:** Evaluates how often a system provides incorrect responses when utilizing either relevant or irrelevant retrieved documents. The score ranges from 0 to 1, with lower values indicating better performance. Noise sensitivity is computed using the user input, reference, response, and retrieved contexts.

$$\text{Noise Sensitivity} = \frac{|\text{Incorrect Claims In Answer}|}{|\text{Claims In Answer}|}$$

- **Response Relevancy:** Assesses how pertinent the generated answer is to the given prompt. Lower scores are given to answers that are incomplete or redundant, while higher scores indicate better relevancy. This metric is computed using the user input, retrieved contexts, and response.

$$\text{Answer Relevancy} = \frac{1}{N}\sum_{i=1}^{N} cos(E_{g_i}, E_o)$$

$$\text{Answer Relevancy} = \frac{1}{N}\sum_{i=1}^{N} \frac{E_{g_i} \cdot E_o}{\|E_{g_i}\|\|E_o\|}$$

, where

  - E_{g_i} is the embedding of the generated question i,
  - E_0 is the embedding of the original question,
  - N is the number of generated questions, which is 3 default.

- **Faithfulness:** Measures the factual consistency of the generated answer against the given context. The answer is scaled to a (0,1) range, with higher values indicating better performance. The answer is considered faithful if all claims made in the answer can be inferred from the context. To calculate this, a set of claims from the generated answer is first identified, then each of these claims is cross-checked with the context to see if it can be inferred.

$$\text{Faithfulness Score} = \frac{|\text{CITATCBIFC}|}{|\text{CITGA}|}$$

, where

  - `CITATCBIFC` - Claims In The Answer That Can Be Inferred From Context
  - `CITGA` - Claims In The Generated Answer

5 Proof of Concept

To lay the groundwork for the MiNI-RAG chatbot, we initiated a systematic approach to data acquisition, processing, and embedding storage. Using the BeautifulSoup library, we scraped HTML data from all accessible pages of the Faculty of Mathematics and Information Science (MiNI) website[4], ensuring the inclusion of comprehensive and relevant content covering faculty-related FAQs. The scraped HTML files were then transformed into text documents. During this transformation, HTML-specific formatting, such as tags and inline scripts, was removed while retaining the most meaningful and informative content. The goal was to optimize text extraction by preserving semantically significant information and eliminating unnecessary noise.

To facilitate efficient processing and retrieval, we employed the RecursiveCharacterTextSplitter method, dividing each document into manageable chunks of text with a chunk size of 1000 characters and an overlap of 200 characters. This method is particularly advantageous as it not only allows for the preservation of context across adjacent segments but also ensures that overlapping sections bridged gaps, enabling coherent understanding during response generation. For embedding generation, we utilized the pre-trained model "Alibaba-NLP/gte-Qwen2-1.5B-instruct," available on Hugging Face. It is a General Text Embedding Model based on "Qwen2-1.5B" LLM model and uses the same training data and strategies as the "gte-Qwen2-7B-instruct" model. This state-of-the-art embedding model provided high-quality vector representations optimized for downstream retrieval tasks, offering robust semantic understanding that effectively captured the nuances of the text. The embedder is hosted on a virtual machine with access to a GPU, enabling efficient processing and embedding generation, and is accessible through a REST API built using FastAPI[5] framework. This model will also be used to embed incoming user queries before searching for the closest chunks in the database.

The vector store was configured using open-source ChromaDB[6] to store the text chunks along with their corresponding embeddings. ChromaDB was chosen for its performance and ability to handle large-scale vector data efficiently, ensuring rapid retrieval of relevant chunks during the chatbot's operation. The vector store is set up locally and is designed to return the five closest chunks to the embedded query, ensuring relevant and contextually accurate responses.

Chunks retrieved from ChromaDB will be used to generate the final response, which will be crafted by the Llama-3.1-8B[7] model from Hugging Face. This model is served using vLLM[8] framework, which is a state-of-the-art inference and serving engine for LLM/VLM. It creates an HTTP server that implements OpenAI's Completions[9] and Chat[10] APIs, ensuring smooth integration and reliable performance. Before generating the response, the model will be provided with a specifically designed prompt, optimized to enhance its understanding and output quality.

The entire project has been designed with accessibility in mind. The user interface was created using Streamlit[11], featuring a chat-like interface to allow users to interact with the chatbot effortlessly. This interface makes it simple for anyone to inquire about information available on the MiNI website, ensuring a user-friendly experience.

For further details, please refer to appendix A, where we present histogram of number of character on pages, architecture diagram alongside examples of test questions and corresponding responses from the chatbot. These responses are compared with verified information sourced directly from the MiNI website to ensure accuracy and reliability.

6 Tools & Equipment:

Hardware: Cloud resources (GCP) with GPU/TPU support for embedding and generation tasks, and Eden (university cluster with GPUs) for additional computational power.

Software: Python (`BeautifulSoup` for scraping, preprocessing, and model integration), Langchain (for RAG pipeline integration), Hugging Face Transformers (for LLM and embedding models), ChromeDB (for vector stores and

[4] https://ww2.mini.pw.edu.pl/

[5] https://fastapi.tiangolo.com/

[6] https://www.trychroma.com/

[7] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[8] https://docs.vllm.ai/en/latest/index.html

[9] https://platform.openai.com/docs/api-reference/completions

[10] https://platform.openai.com/docs/api-reference/chat

[11] https://streamlit.io/

retrieval), FastAPI (for deployment of embedding model as an API, if needed), VLLM for LLM serving and inference and Streamlit (for creating the user interface).

References

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia, May. Association for Computational Linguistics.

Subash Neupane, Elias Hossain, Jason Keith, Himanshu Tripathi, Farbod Ghiasi, Noorbakhsh Amiri Golilarz, Amin Amirlatifi, Sudip Mittal, and Shahram Rahimi. 2024. From questions to insightful answers: Building an informed chatbot for university resources. *arXiv preprint arXiv:2405.08120*.

Krzysztof Ociepa, Łukasz Flis, Krzysztof Wróbel, Adrian Gwoździej, and Remigiusz Kinas. 2024. Bielik 7b v0.1: A polish language model – development, insights, and evaluation. *arXiv preprint arXiv:2410.18565*.

Szymon Ruciński. 2023. Krakowiak-v2-7b. `https://huggingface.co/szymonrucinski/krakowiak-v2-7b/`.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of retrieval-augmented generation: A survey.

A Appendix

In this appendix we present architecture diagram and and some test questions and responses from chatbot compared with truth information from the MiNI website. We also present histogram showcasing number of characters in different pages.

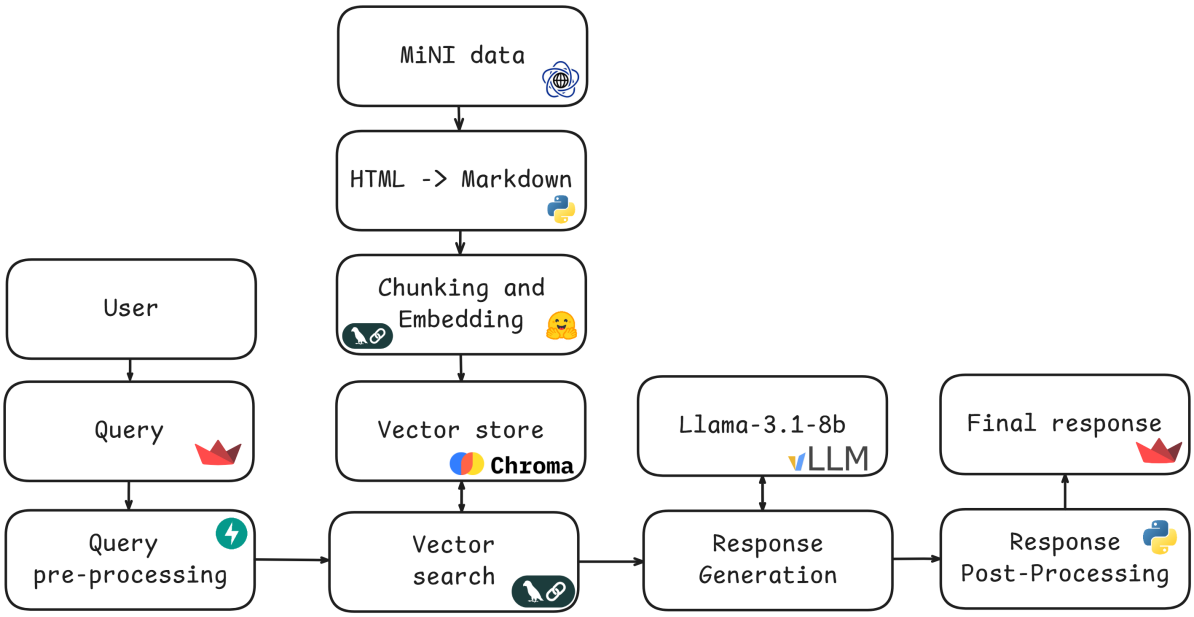

Figure 1: Architecture diagram.

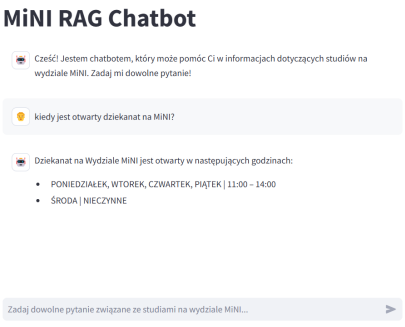

Figure 2: Information provided by our chatbot.

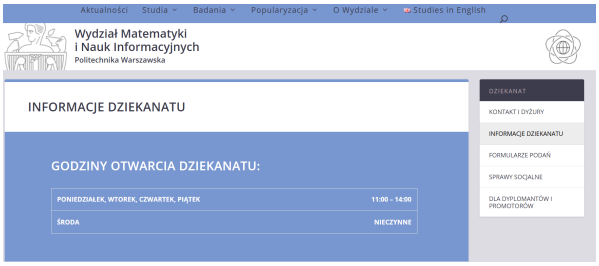

Figure 3: Information from MiNI page

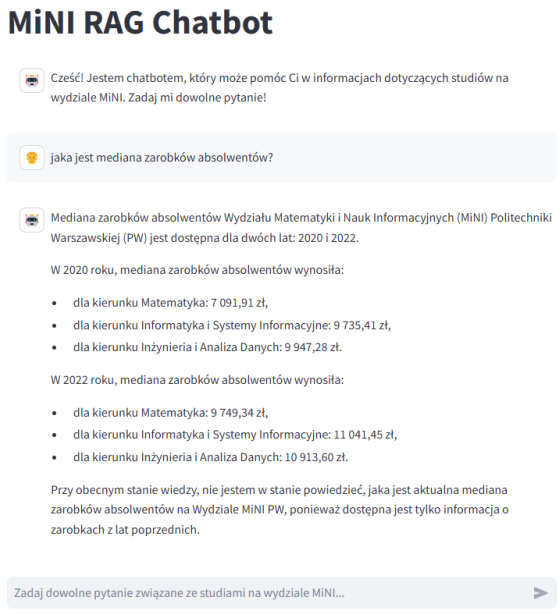

Figure 4: Information provided by our chatbot.

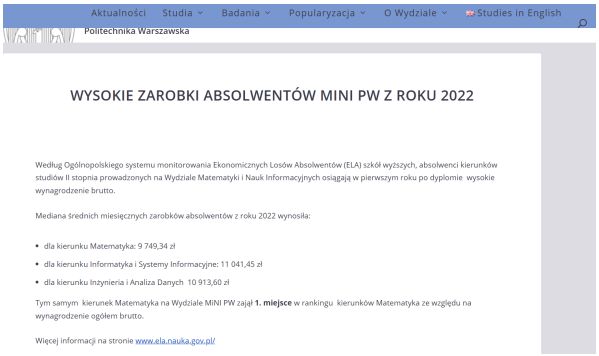

Figure 5: Information from MiNI page

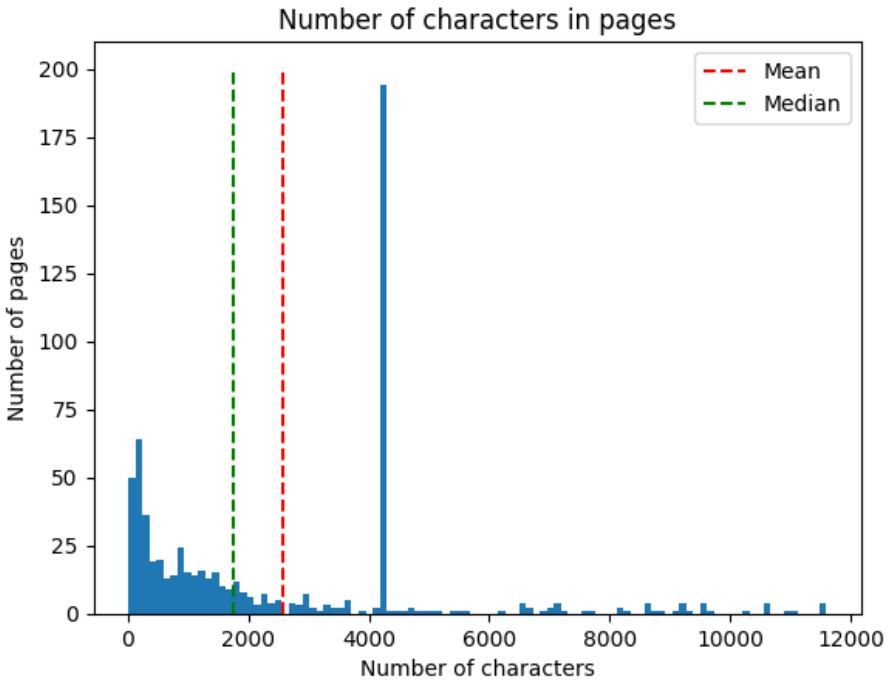

Figure 6: Histogram of characters