Automatic Summarization System

Christian Faccio[1] *, Elena Lorite[1] †, Rebeca Piñol Galera[1] ‡and Paula Frías Arroyo[1] §

[1]University of Alicante

Abstract

In this project we developed an automatic summarization system, which is able to effectively summarize the information present on a set of documents. The process is divided in two steps: information retrieval, done using a non-neural approach with RegEX and a neural approach with transformers, and generation, completed with an encoder-decoder model. We finally analyzed the effectiveness of both approaches, showing that modern state-of-the-art transformers are able to capture more information than standard information retrieval methods, but at the cost of efficiency and possible problems like hallucinations.

Introduction

Summarizing the text contained in a document is a difficult task even for people. It necessitates the individuation of the most important topics, then the selection of the most important part of them and finally the generation of a smaller text starting from that. **Automatic Summarization Systems** are employed to help in this task: they should be able to summarize the text of the documents autonomously or with limited and small human help like in this case. It is a task which divides in two parts: the first is the individuation and extraction of the most important information, while the second is the generation of new text starting from that. In our case, the information to be extracted is defined beforehand by us, so the first architecture just needs to find the most relevant parts of the text with respect to that. The developed code and relative data can be found in our GitHub repository.

Data

The documents used in this project are scholarships for students published in the Boletín Oficial del Estado (BOE) and can be found in the `docs/` folder. They are 5 documents in total and are mostly similar with respect to the format.

The information extracted is detailed and specific:

- academic year;
- total budget;
- income thresholds;
- eligible programs;
- scholarship components;
- application period;
- academic requirements.

We believe that this data can represent most of the useful information contained in the documents and a special consideration must be done regarding the type of data extracted, since it is not just common text but comprises ranges, dates, prices and structured information. Given this variety of data types, the information retrieval part had to be specifically tailored for this task, as we will describe in the next section. Finally, the extracted information has been written on a JSON file in a structured manner, which can be found in the `output/` folder.

Architecture

We divided the whole architecture in two components: the *information retrieval (IR)* part and the *generation* part.

Information Retrieval

This was the part in which we spent most time on. Without correct and reliable data, the summarization cannot be done, so this is probably the most important part of the system. There are different methods to be used here, but the most common are the use of *regular expressions*, which are not flexible at all but can extract information with special format like dates, the use of *similarity search* between a query and the chunks of the text, which is more flexible but contains hidden difficulties like the choice of the chunking method, and finally the use of a *transformer* model, which is the state of the art and is flexible, even if costly and complex.

Even if usually a transformer architecture is the best choice, it may not be for every situation. In fact, the best trade-off between performance and efficiency depends on the caractheristics of the problem at hand. In this case, chunking the text and then use similarity search between a query and them was not

*christianfaccio@outlook.it
†elenalorite@gmail.com
‡rpg80@alu.ua.es
§pfa13@alu.ua.es

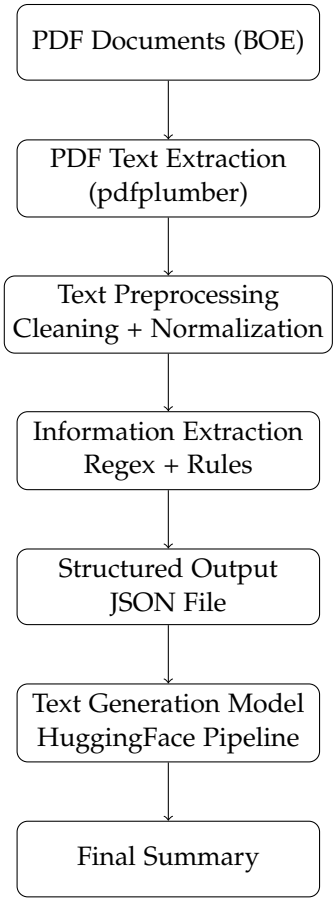

Figure 1: *System Architecture*

the best approach. We, in fact, tried to use a semantic chunker to split parts of the text which are coherent with respect to a speech topic, saved them in a vector store and ran similarity search with the topics of interest. The results were good, the data extracted were coherent with the query, but the main problem was that this way was impossible to extract detailed and structured information as we needed to. This necessity comes from the fact that the documents at hand contain information regarding scholarships, so a summary of them should include detailed information which are helpful for a possible candidate, and that include dates, ranges and mostly numbers. Thus, a better approach in this specific case is the use of regular expressions.

We would like to emphasize that this method was possible just because the documents are similar in writing and format, thus the lack of flexibility of the method. But, given that we needed structured and specially formatted information, this was a good approach. The results are shown in the `info.json` file and are pretty good indeed, even if in some cases we were not able to extract all the data. Unfortunately, with small text variations this method stops working correctly.

We also implemented an alternative approach with a transformer, to see and compare the performance with the former method. With this approach we wanted to evaluate whether a transformer-based model could reduce the amount of the manual logic while improving flexibility and robustness. Unlike the previous method, which relied a lot on rules, regular expressions and specific heuristics, the LLM-based system leaves most of the semantic interpretation to the model itself. First the raw text from the PDF files is extracted and then it is segmented by articles with the use of regular expressions. After, each relevant section is provided to the LLM with a specific prompt for each section. The extracted data from all documents are then combined into a structured file: `info_mistral.json`. This approach provided several advantages like higher flexibility and better semantic understanding. However, this approach also has certain downsides: hallucination risk, higher computational cost and output variability. To evaluate the impact of model size on performance, we experimented with several open-source LLMs ranging from 3B to 70B parameters (3B, 8B, 14B, 24B, 30B and 70B). As expected, larger models generally produced more accurate and consistent structured outputs, with the 70B model (Llama 3.3) achieving the best performance. However, when considering cost per token, inference latency and overall efficiency, the 24B parameter model (Mistral Small 3.2) offered the best balance between performance

and computational cost. Smaller models were capable of completing the task, but their outputs showed a degradation in structural consistency and precision. Based on these experiments, models around 24-30B parameters appear to be the best ones for this type of structured extraction task. All inferences were performed through a remote API (OpenRouter), rather than running the models locally. This inference choice simplified experimentation across multiple model sizes and use of the chosen model. Overall the LLM based approach proved to be more adaptable, if needed, and easier to extend to new formats, at the cost of increased computational complexity and potential reliability issues.

Generation

The summary generation pipeline was implemented using open weight instruction tuned language models accessed through Hugging Face Transformers and executed locally. In order to ensure that all group members could run the system on their own hardware, the selected models were limited to approximately 1.7B–3.5B parameters, requiring at most around 6GB of VRAM. The models used in the final version are Phi-3.5-mini-instruct, SmolLM2-1.7B-Instruct, and Qwen2.5-3B-Instruct. These models were chosen because they provide a good balance between computational efficiency and instruction following performance, while remaining lightweight enough for local execution.

Phi-3.5-mini-instruct is a compact but highly optimized instruction-tuned model designed for strong reasoning and controlled text generation. Qwen2.5-3B-Instruct, with approximately 3 billion parameters, is a more recent instruction aligned model known for stable multilingual and structured generation capabilities. SmolLM2-1.7B-Instruct represents an even smaller architecture, included to evaluate how very lightweight models perform under the same prompting strategy. By comparing these three models, we analyze how differences in parameter scale and architecture influence summary quality.

The summaries were generated from structured JSON data. Each JSON entry was transformed into a designed prompt that explicitly instructed the model to act as an academic writer, produce a structured summary of 200–250 words, and avoid inventing information. The generation process used controlled parameters, including a moderate temperature (0.4), nucleus sampling (top-p 0.9), and a repetition penalty to reduce redundancy. These settings aim to balance determinism and linguistic naturalness and limit variability across runs.

In addition to summary generation, the pipeline incorporates a factual consistency analysis step. A multilingual Natural Language Inference model based on mDeBERTa-v3 is used to compute a score between the original structured data (treated as the premise) and the generated summary (treated as the hypothesis). This hallucination score serves as an observational metric to estimate the alignment between the JSON file and the generated summary. We expect Phi-3.5 and Qwen2.5-3B to produce the most coherent and factually consistent summaries due to their stronger instruction tuning, while SmolLM2-1.7B may show more variability given its smaller parameter size.

Evaluation

Conclusions