IRNLP Project Report

Pierfrancesco Martinello, Giuseppe Termerissa

Intro

The objective of this project is twofold: we wish to retrieve a high amount of webpages whose domain is www.unipa.it by using web crawling techniques, then feed the obtained data to a Retrieval-Augmented Generation (RAG) pipeline with the intent of building a system that behaves in a similar fashion to chatbots, starting from two pretrained LLMs: Zephyr-7B and LLaMA-3-8B. The system is then asked several questions regarding the University of Palermo, returning appropriate answers that can fit real-world scenarios.

The project is available at the following repository: Github.

The directory structure is as follows:

- `output`, containing output files generated by the crawler
- `src`, containing source files for the crawler
- `test`, containing the crawler's unit tests
- `rag_v2.ipynb`, containing the RAG's pipeline

Execution Instructions

Executing the crawler is only a matter of running the file located in `src/crawler.py`, optionally changing the parameters inside the main block. A safety feature has been implemented: by hitting CTRL-C only once, the script will be halted and the results found so far will be saved.

In order to execute the RAG notebook without errors, a few things are necessary.

Due to the gated nature of the model Llama 3, it is necessary to ask permission from the owner for its usage. By simply following this link, it is possible to fill a form and ask for permission. Usually it will be granted in a matter of minutes. If the access has been granted already, it will be promptly displayed.

It is necessary to use a HuggingFace key as a secret. From the following link, go to “Access Token” and then click “New Token”. After giving it a significant name and specifying “Fine Grained (custom)” as its type, it is possible to generate it.

Then, by clicking “Set Permissions”, a few setup steps are necessary:

1. In the Repositories permissions section it is necessary to select “meta-llama/Meta-Llama-3-8B” and check the box “Write access to contents/settings of selected repos”.
2. In the User Permissions/Collections sub-section is necessary to toggle “Write access to all collections under your personal namespace”.
3. In the User Permissions/Inference sub-section all options must be on
4. In the User Permissions/Repos sub-section the option “Write access to contents/settings of all repos under your personal namespace” must be selected.
5. Finally, the key can be stored in Google Colab under the name ‘HF_TOKEN’.

If Google Colab is used, the first cell in the RAG notebook must be executed alone, and after it ends it is necessary to restart the session. After that all other cells can be executed, and executed only once, otherwise restarting again the session is necessary.

Crawling

The first part of the project involves the creation of a Crawling system in Python, using libraries such as BeautifulSoup to handle the extraction of textual information from webpages.

The system has been created based on how Unipa’s website handles its textual web pages: text is consistently contained inside <article> tags, each containing <p> (paragraph) tags, finally containing the textual data we wish to extract. Metadata such as each page's links and titles is kept and used as keys for the dataset’s rows.

Crawling is limited to websites with a unipa.it domain: this is to avoid the crawler from walking to external pages, possibly downloading uninteresting pages. Additionally, checks are used to avoid non-html pages from being downloaded, which would entail a completely different kind of processing.

The crawling mechanism is limited to the main domain (unipa.it), hence avoiding subdomains (such as idp.unipa.it, a3.unipa.it, etc.)

The model performs a small amount of cleaning of the retrieved pages, but most of the cleaning is actually outsourced to the RAG's pipeline, where most of the actual preprocessing is done.

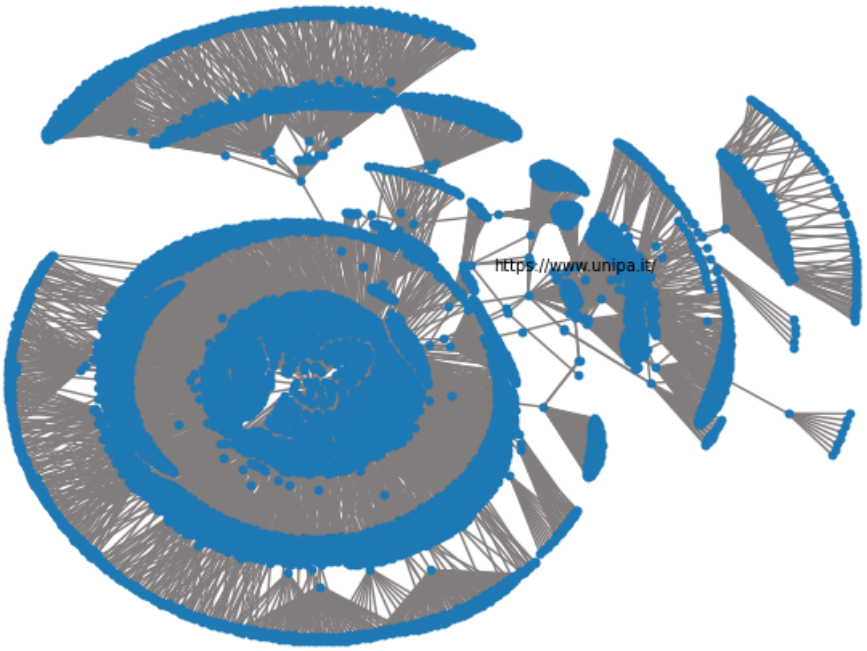
Last, the crawler supports several features:

- two walking strategies (BFS and DFS)
- parameter for maximum depth
- parameter for maximum number of visits
- support for topology plotting by using NetworkX.

Dataset

The dataset used in training has been constructed by iterating in a BFS fashion for a maximum depth of 30, starting from https://www.unipa.it/. Crawling has been performed until every page in the set has been visited, fetching an exhaustive list of articles regarding the University.

The final dataset is composed of 6338 rows, 2776 after cleaning, where "garbage" data points containing only urls or footers have been filtered.



Dataset topology, offering a vague idea of how the crawler moved through the websites

RAG

A mix of Haystack and HuggingFace has been used to construct the RAG pipeline and to access the required pretrained models. As stated before, the models that were used are Zephyr-7B and LLaMA-3-8B (the latter requiring a particularly set up access token, as specified below).

The pipeline is roughly structured as follows:

1. **Preprocess the data**: remove "title" timestamps, remove empty documents, trim leading/trailing whitespaces, remove documents shorter than 200 characters;
2. **Perform indexing**: split each document two sentences at a time, compute the embeddings and write them to a document store;
3. **Prepare RAG pipeline**: setup the chosen model, setup a prompt format, setup a text embedder for queries and a retriever that will fetch documents from the document store. This is done once per model;
4. **Perform queries** on the RAG.

Results

Several evaluation frameworks have been considered, such as RAGAS, Galileo and Arize. However, issues have arised with all of these, namely the fact that they're either not free or locked behind an OpenAI API key (also not free). Also, several attempts have been made in order to avoid the dependability from OpenAI LLMs, but yielding no result. For these reasons, no framework has been used for evaluation.

Additionally, several renown Information Retrieval evaluation measures have been considered, but these still present problems:

- Precision entails the ability to generate several retrievals for each query, while this pipeline only allows one answer per query;
- Recall entails knowing which documents are relevant in the entire set. Furthermore, the aforementioned inability to output more than one answer per query does not allow its calculation;
- Precision-Recall curve is typically implemented in the case of systems where not every document has been retrieved, but a combination of the above problems still makes this an unfeasible metric.

A possible alternative could have been to manually label each document, but the dataset is unfeasibly big to be labeled in such a short amount of time. \

Thus, no real empirical evaluation phase has been realized for this project, and the only kind of evaluation that has been done is by asking questions to the RAG and comparing the results with few known ground-truth answers, in a manual way.

From our observations, both Zephyr and Llama tend to work similarly: both of them tend to correctly answer the proposed questions, with results that might be applicable in a real-world scenario. Examples are shown at the end of this notebook.