EULA: Embedding Utilizer Legal Assistant

Nicola Debole
University of Trento
nicola.debole@studenti.unitn.it

Abstract

In legal frameworks, the task of searching through vast amounts of data to locate pertinent judgments, which can aid in building a defense or gaining insights into legal precedents, is frequently required. Also the general public lacks the necessary tools to conduct independent research, unless they possess legal expertise. Consequently, the concept of developing a legal assistant capable of retrieving documents relevant to specific cases emerged. This report aims to present a series of experiments focusing on this subject, along with a functional demonstration of EULA. The proposed system employs the BM25 algorithm for retrieving matching documents, and subsequently utilizes semantic similarity between sentences to refine the search process from the retrieved documents.

1 Introduction

The project was developed by a single person, but for the sake of the report no singular person will be used and substituted with *we*.

The objective of this project is to effectively retrieve relevant legal cases from a database of past judgments based on user queries. To achieve this, we needed to face many different obstacles especially for collecting the data necessary for the project since no public available website was providing those kind of judgements.

Initially, we developed a web scraper capable of downloading up-to-date sentences from https://www.giustizia-amministrativa.it/ , as it was crucial to work with real and current data. Subsequently, we performed preprocessing on the acquired data to ensure it was in the right format for the next steps.

Once we obtained a viable dataset, we experimented with different models. Firstly, we trained a BERT model on the preprocessed data, followed by training a LONGFORMER model. Finally, we employed a combination of BM25 and a sentence-transformer for document retrieval from the database, without the need for fine-tuning.

In this report, our primary focus will be on the findings and results of our experiments. Additionally, we will address the challenges we encountered during the project.

At the end in 9 we will provide all the links for extensive results and datasets to reproduce this work.

2 Readings

Before developing any solution, since the topic was already covered in some published papers, it was interesting to read how the problem was approached by experts in this field. Here are some readings in no particular order:

- (Junior et al., 2019)
- (Locke and Zuccon, 2022)
- (Yu et al., 2022)
- (Bhattacharya et al., 2022)
- (Sun et al., 2023)
- (Hwang et al., 2022)
- (Nguyen et al., 2022)
- (Zhang et al., 2023)
- (Louis et al., 2023)
- (Santosh et al., 2023)

We briefly skimmed through some of the topics to gain a general understanding of how things work, while others caught our attention and delved deeper into them.

Despite giving useful information we decided to still have a personal and probably naive approach to the problem for two main reasons:

Model	max tokens	Language
bert-base-multilingual-cased	512	multilingual
allenai/longformer-base-4096	4096	english
efederici/sentence-BERTino	512	italian

Table 1: The three models we used in order to experiment and find a working solution for the problem described in this report.

- The goal of the project is to learn how to build, in this specific situation, an information retrieval model and pipeline. Since our personal knowledge on the topic is not that advanced, we preferred to start from the ground up instead of just implementing SOTA models without understanding the inner workings.

- Second, no similar work was present in the literature with our requirements for the solution (i.e. to be in Italian, work on legal domain, unlabeled dataset, open-source)

3 Crawling and Scraping

Actually in order to create a dataset we had to resort to code a web crawler and scraper to first broswe to the website [1] then input the correct terms and parameters for the query. The website would then generate a list of links redirecting to different judgements: at this point the job was to collect all those web links and move to the next page to generate new links, and so on.

The step two was to, from all the collected links, download the pdf file, converting it to a txt file and store all of them with the attached metadata like the location, the id of the judgement and the link.

This was all done using a custom created script in Python that uses a real chrome instance to navigate the website, and this was necessary because the website was dinamic and all the links were generated by a Java script. All the code is in this repository[9], though it is not really user-friendly since it we have little to no expertise in this kind of kind of tasks and there was no time for improving the re-usability of the mentioned code.

4 Dataset

Here we provide a short list of the datasets used (that we generated or already available online). The ones that cannot be found on the web are provided in the **all_datasets.zip** file [9].

The working demo that we will cover later is using the Judgements dataset [4.2] but a brief mention on all the ones we used.

4.1 Web links

This contains all the links of the judgement pdfs and they are stored as a csv file. Every batch contains 3000 links.

```
    # Web links dataset location
    all_datasets/crawler/dataset/
```

4.2 Judgements

From the Web links dataset we downloaded the pdfs and converted them to a txt format. There are two folders:

- pdf where the files are stored as .txt files and with no preprocessing

- raw_pdfs where the files have been preprocess including metadata and stored as a .pickle file

Eventually, only the raw_pdfs folder is used in later stages.

```
    # Judgement texts
    all_datasets/crawler/datasets/
        raw_pdfs
```

4.3 mc4_legal

This dataset was downloaded from Hugging Face and the complete name is **joelito/mc4_legal**. It has many already preprocessed legal documents and available in many different languages. It was used to fine-tune the longformer model because fine-tuning the model on the same dataset from where the documents were going to be retrieved was believed as a cheat (and maybe not optimal for generalization purposes).

4.4 Other folders

All the other folders contains data that has been processed in different ways and it is used as input for the different models. For example the Masked Language Modeling dataset that we created for training the LONGFORMER model is accessible at this location:

```
    # Masked Language Modeling dataset
    all_datasets/longformer/MLM/
```

there are different folders and each one of them has a different variation of some parameters that drive the generation of the dataset (e.g. number of masked tokens, padding or no padding, etc.). Explaining what every folder is for is out of the scope of this report since it would take a lot of time and give no useful information since most of them are just discarded work that generated bad results. Also they all come from this starting point [4.2] that we mentioned above and have just slight variations.

5 Models

As visible in the table [1] we used 3 different models. They are presented in chronological order of use.

5.1 BERT

This was the first model used because of the simplicity of the model, it was pretty well documented and tested (Devlin et al., 2019). We used Hugging Face to import the version for MLM[1]. Amongst the languages supported there is Italian so it was able out of the box to understand the language correctly and the original idea was to fine-tune it (even though the amount of data we had available was enough to train it from scratch) in order to let the model get some sort of understanding of the legal domain. Unfortunately this did not yield good results and from checking the loss the model was not converging at all. That is the reason this model was soon discarded. The reason behind it could be that, because of the nature of the documents (long and information-scarce) we needed more context (more tokens) to make some "logical" connection. After inspecting intensively the dataset, there were times that a single chunk of 512 tokens could contain only proper names of people involved in the specific situation. So the solution was to move to a more capable model in terms of context.

5.2 longformer

This was the second attempt and the pipeline remained basically the same:

1. Prepare a dataset for the MLM task: following (Devlin et al., 2019) we masked 15% of the tokens (even though played a little with the values) and between that percentage a small fraction was kept as the same token and another fraction was changed with a random token. This is to make the model more resilient and learn from the context.

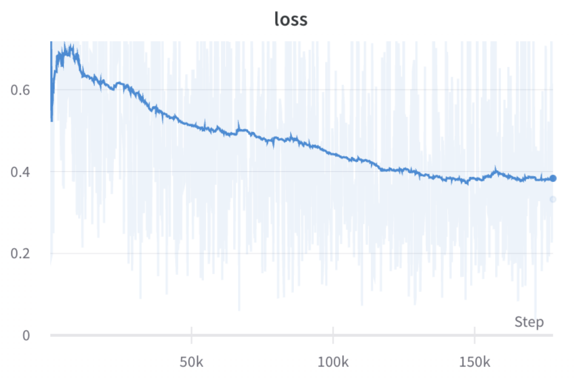
2. Train the model: check the loss to see if the model is converging

3. Evaluate the results

[1]Masked Language Modeling

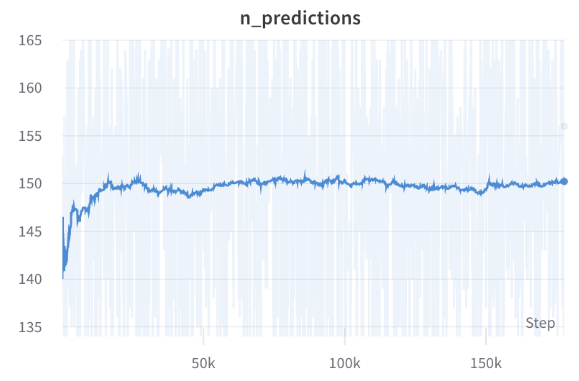
Now since the max input tokens allowed was increased to 4096, we just had to create a dataset matching that size. If from one point of view the increased context size was really good the model does not support Italian and so had to be trained for more time. Also, since the bigger model, the more time it would take to complete a train step and so a full epoch (100000 samples) was never achieved. More details on the model used can be found on the paper (Beltagy et al., 2020) or on Hugging Face allenai/longformer-base-4096. From ?? it is possible to see that the model is pretty good at predicting the masked tokens even without training, and the whole training session lasted 1d 16h 40m 16s.

Figure 1: Loss during training of the best model version.



Since the input text was not always reaching the limit of the 4096 tokens, using the standard way to create the dataset would mean that the number of masked tokens would vary and also the loss value would fluctuate a lot. We had the idea to fix the number of masked tokens, and this was done using an easy function that computes the probability to have in order to have a set amount. So during the dataset creation process, based on the input size the probability changes to keep the number of masked tokens close to the desired value. As you can see from the Figure 2 down below, the number of masked tokens stays very close to the range $[130, 170]$ as we set 150 as the desired numer of masks per input. Relative to the previous dataset, where the number of predictions was changing by a lot in the range of $[0, 600]$, our method provided a more stable dataset. This was done because originally in the train loss graph there were a lot of spikes resulting from a steep drop or increase in the number of predictions to calculate.

Figure 2: Number of predictions per input i.e. the number of masked tokens passed to the model.



5.3 sentence-BERTino

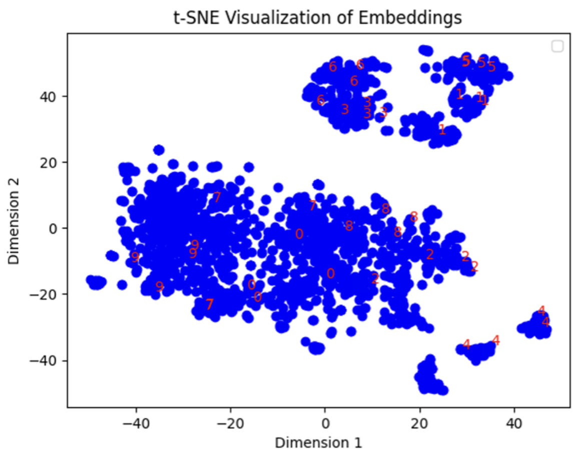
Briefly, this is a sentence-transformer model that has been fine-tuned on Italian available on Hugging Face efederici/sentence-BERTino. After the suggestion to use a sentence transformer to compute the similarity between the query and the documents instead of just computing the vector embedding of whole chunks of the document, and since the previous models were not working, we decided to implement a working solution eventually. This was a very different process: no training was involved and now the problem changed to produce a working pipeline able to segment each document in the dataset, compute the embeddings and then save them in a optimize data structure in order to have a fast retrieval. We will talk more in-depth about this process in the EULA [7] section.

6 Results

We will present here only the results obtained from the longformer model due to not having a way to compute scores and the only way to analyze the model was to actually try to explain the embedding space. The most important result is that this implementation was not able to correctly retrieve any document closely related to the query. Instead BM25[(Robertson et al., 2009) was getting every time correct documents if not exactly the one we were looking for. This should have been expected from the beginning, especially after reading (Rosa et al., 2021) where it has been shown that a simple BM25 approach is a strong baseline for the legal case retrieval task, also hard to beat for dense retriever.

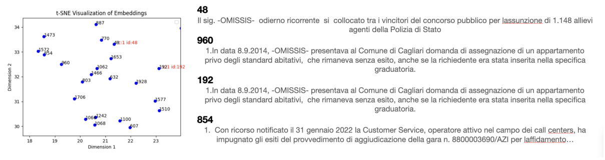
We run both a k-means clustering algorithm and a t-SNE algorithm to try to understand and visualize the embedding space. From this Figure we can

Figure 3: Visualize the embedding space in 2D using t-SNE and the different clusters arrangement using k-means.



assess that the model can classify the documents but the problem is that the classification the model learned is not the one we would like to use in any work application. This kind of classification makes no sense to a human person and it is hard to even understand if it has any logic. But by further analyzing each separate cluster, we can show some examples:

Figure 4: This cluster is about "Gara, Graduatoria", the one highlighted in red are just to show some elements of cluster 1 but all the documents shown are actually also in the same cluster. On the right there are 4 snippets of text contained inside some of the documents.



48
Il sig. -OMISSIS- odierno ricorrente si collocato tra i vincitori del concorso pubblico per lassunzione di 1.148 allievi agenti della Polizia di Stato

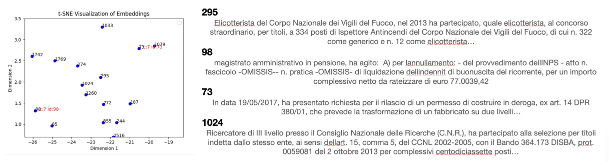
960
1.In data 8.9.2014, -OMISSIS- presentava al Comune di Cagliari domanda di assegnazione di un appartamento privo degli standard abitativi, che rimaneva senza esito, anche se la richiedente era stata inserita nella specifica graduatoria.

192
1.In data 8.9.2014, -OMISSIS- presentava al Comune di Cagliari domanda di assegnazione di un appartamento privo degli standard abitativi, che rimaneva senza esito, anche se la richiedente era stata inserita nella specifica graduatoria.

854
1. Con ricorso notificato il 31 gennaio 2022 la Customer Service, operatore attivo nel campo dei call centers, ha impugnato gli esiti del provvedimento di aggiudicazione della gara n. 8800003690/AZI per laffidamento…

It could also be that the clusters that make sense actually contains more outliers that we were not

Figure 5: This cluster instead seems not to have any logic and contains documents of very different topics. On the right there are 4 snippets of text contained inside some of the documents.



295
Elicotterista del Corpo Nazionale dei Vigili del Fuoco, nel 2013 ha partecipato, quale elicotterista, al concorso straordinario, per titoli, a 334 posti di Ispettore Antincendi del Corpo Nazionale dei Vigili del Fuoco, di cui n. 322 come generico e n. 12 come elicotterista…

98
magistrato amministrativo in pensione, ha agito: A) per lannullamento: - del provvedimento dellINPS - atto n. fascicolo -OMISSIS-- n. pratica -OMISSIS- di liquidazione dellindennit di buonuscita del ricorrente, per un importo complessivo netto da rateizzare di euro 77.0039,42

73
In data 19/05/2017, ha presentato richiesta per il rilascio di un permesso di costruire in deroga, ex art. 14 DPR 380/01, che prevede la trasformazione di un fabbricato su due livelli…

1024
Ricercatore di III livello presso il Consiglio Nazionale delle Ricerche (C.N.R.), ha partecipato alla selezione per titoli indetta dallo stesso ente, ai sensi dellart. 15, comma 5, del CCNL 2002-2005, con il Bando 364.173 DISBA, prot. 0059081 del 2 ottobre 2013 per complessivi centodiciassette posti…

able to find since we just scrubbed 4-6 documents per cluster.

7 EULA

The last attempt consist in the following pipeline:

1. Have the original dataset containing all the judgements segmented by sentences using the haystack preprocessing tool.
2. Each sentence is embedded in a vector by the model, and that all the vectors are stored in a pickle file named as the document ID.
3. The user write a sentence, that is embedded run-time by the model.
4. BM25 retrieves a N amount of documents matching the query that the user wrote, and outputs their IDs.
5. For each ID returned, the vectors of that document are accessed and the cosine similarity is computed (between the sentence embedding and the query embedding).
6. The min, max and mean values of the cosine similarity are saved for each of the N documents BM25 retrieved.
7. Based on the user selection, the N documents are now ranked based on the min, max or mean value of the cosine similarity.

We decided to add BM25 to make the query faster, otherwise it would have taken ages just to compute all the similarities for a single query. The main problem was not to have an optimized index, due to the inability to use one such as FAISS[2]. Time was running out and despite haystack claimed that FAISS could work out of the box within their library, it did not. So the implementation for storing the embeddings and searching was the easiest but slowest and unoptimized one.

Moreover, since BM25 is such a strong baseline it would be wise to use it anyway in the pipeline, and then use a dense retriever to handle fine grain document selection. The model used accepted maximum 512 tokens, and from the dataset analysis there are at least some sentences that exceed this limit but it should not be a problem since the average we computed is around 500 tokens.

The results obtained are acceptable, most are relevant even if the sentence-transformer sometime classify two sentences to be similar while in reality one is just a single 5 words line with little to no meaning (e.g. Art. 1 del Codice Civile) and that makes the model retrieve wrong documents especially when using the maximum value of the cosine similarity score to rank them. Also another problem is that the dataset creation was very slow (since we needed one for BM25 that splits the documents per word and one for the sentence-transformer that splits the documents per sentences). This forced us to create only two partial datasets that do not exactly match so it is possible that after retrieving documents from BM25 some could have been processed by the model to compute the embedding vectors and so are discarded (while they may be very good results). Despite being very annoying, it is easily fixable by just letting the script run for the time required to generate both datasets. In any case, you can try the working demo we provided in the EULA Github [9] repository and following the instructions from the README file.

8 Conclusion

For this project many different solutions were used, most of them did not work but actually this is the process of research. Moreover, by attempting such various methods we learned a lot more than just by trying just a single successful implementation. In particular, the task a model is trained for is a topic way more "deep" than we thought in the sense that having a model that output the most probable next word or does masked language modeling is much more different than expected. And also understanding that difference is of paramount importance. Also datasets need to be designed for a specific task and they do not always work out of the box. A last important lesson is about tokenizers, often overlooked, that actually are the foundation of the model: using them wrongly can cause issues.

[2]https://github.com/facebookresearch/faiss

9 Links

- Github Repository crawler
- Github Repository EULA
- Weight and Biases Project Page
- all_datasets.zip

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Paheli Bhattacharya, Kripabandhu Ghosh, Arindam Pal, and Saptarshi Ghosh. 2022. Legal case document similarity: You need both network and text.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Wonseok Hwang, Saehee Eom, Hanuhl Lee, Hai Jin Park, and Minjoon Seo. 2022. Data-efficient end-to-end information extraction for statistical legal analysis.

Ademar Crotti Junior, Fabrizio Orlandi, Declan O'Sullivan, Christian Dirschl, and Quentin Reul. 2019. Using mapping languages for building legal knowledge graphs from xml files.

Daniel Locke and Guido Zuccon. 2022. Case law retrieval: problems, methods, challenges and evaluations in the last 20 years.

Antoine Louis, Gijs van Dijck, and Gerasimos Spanakis. 2023. Finding the law: Enhancing statutory article retrieval via graph neural networks.

Ha-Thanh Nguyen, Manh-Kien Phi, Xuan-Bach Ngo, Vu Tran, Le-Minh Nguyen, and Minh-Phuong Tu. 2022. Attentive deep neural networks for legal document retrieval. *Artificial Intelligence and Law*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Guilherme Moraes Rosa, Ruan Chaves Rodrigues, Roberto Lotufo, and Rodrigo Nogueira. 2021. Yes, bm25 is a strong baseline for legal case retrieval.

T. Y. S. S. Santosh, Philipp Bock, and Matthias Grabmair. 2023. Joint span segmentation and rhetorical role labeling with data augmentation for legal documents.

Zhongxiang Sun, Jun Xu, Xiao Zhang, Zhenhua Dong, and Ji-Rong Wen. 2023. Law article-enhanced legal case matching: a causal learning approach.

Weijie Yu, Zhongxiang Sun, Jun Xu, Zhenhua Dong, Xu Chen, Hongteng Xu, and Ji-Rong Wen. 2022. Explainable legal case matching via inverse optimal transport-based rationale extraction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Ruizhe Zhang, Qingyao Ai, Yueyue Wu, Yixiao Ma, and Yiqun Liu. 2023. Diverse legal case search.