Retrieval Augmented Generation for Legal Journals

Sheeba Moghal, Shingai Nindi, Powell Sheagren

### Introduction

In a now infamous filing last June, lawyers representing a man who claimed to have suffered physical damages during a plane flight submitted the first ever AI generated claim to court. The claim used pre-existing cases to show precedent and all together seemed to have a coherent argument. There was only one problem, the judge and opposition couldn’t find the cases and, even after following up with ChatGPT, they were unable to prove their existence or validity. They claimed to be using the AI tool as a search engine, and it cost them the case and some hefty fines for negligence. In this project, we hope to provide an alternative to legal malpractice by using a retrieval augmented generation (RAG) model to build on a pre-existing LLM such that accurate and correct information can be found through searches.

Law is just one of the many fields in which LLM ‘hallucination’ can cause tangible harm and lead to negative outcomes for users. The LLMs never claimed to be able to be experts on all topics but they do offer a framework for others to build specialized systems to retrieve specific information rather than simply predict the next word in a series. Since so much of legal work is research into precedent, we thought it would be an apt place to apply RAG so that a future model can save lawyers time while not costing them a drop in accuracy.

First, we will discuss the data that we sourced to complete this task, then we will discuss the methodology behind our models and the tools we used. Afterwards, we will show the results of our models and test their validity. Finally, we will return to the initial problem of researching legal cases and show the efficacy of the method to complete that task.

### Methodology

#### Data Sources

For our project, we were lucky that there are quite a few pre-existing repositories of legal filings and documents. This is mostly due to the fact that lawyers need to be able to access the records as easily as possible to ensure they can find relevant sources and we are lucky that this ease of access has led to a few important corpus’ which we used. The first of which is… Next we used… These initial datasets were modified from their original structure to fit our models, allowing them to be called upon and represented in a way that will improve user experience in the final project. While we did consider using multiple sources of data at the same time, it became clear that the merging of the data would be more tiring although we are not ruling it out for future more refined models with increased computational power. Speaking of which.

#### System

In order to run the calculations, we used both our local machines and a cluster on Jetstream 2 through a jupyter notebook environment. This cluster included 18 cores of cpu with 64 GB of memory. 250 of storage, and one GPU which we did not end up utilizing. The cluster was helpful in getting the program to run faster than it would have on a local machine and allowed us to use larger amounts of data. We are grateful to have had the opportunity to use it to run our programs. Speaking of which.

#### Programming

All of the programming was done in python with traditional data wrangling and cleaning packages like pandas and numpy. When working specifically on the model, experiment one utilized BERT and more specifically colBERT as a transformer in a variety of ways which will be explained later.

### Experiments

We started by running three different approaches for the retrieval augmented generation to compare and contrast validity and results: ColBERT, FAISS, and TF-IDF. After some work, we decided to scrap the TF-IDF approach and focused on the other two. I will now break this section description into two to discuss the methods and processes we used for each.

#### ColBERT:

We began by importing data from [REDACTED] and standardizing it for efficient querying and retrieval in the future. This mostly involved managing the metadata so that formatting was in an ideal format and documented properly. The next step involved determining what a query would look like, what questions to ask, and what format the responses would be in. We decided to create multiple fields to look up different aspects such as Name, Legal Abbreviation, Date, Jurisdiction, and a custom option. The query was then preprocessed by cleaning the text and pulling out specific relevant information. Next we created a system that queried the data based on the input through matching dates, names, or other information referenced.

Given this initial prompt, we tokenize and embed it using HuggingFace’s bert-base-uncased which also includes a context step and relevancy fine tuning. This transforms the query into a dense vector representation which is fed into a ColBERT (Contextualized Late Interaction over BERT) model which computes cosine similarity with other documents to ensure relevance and semantic alignment. With this dense retrieval system in place, we needed to add a generative aspect to it to return useful information not just a document number. In order to do this we used BERT traditionally, conditioning on the documents found and using it for an encoder-decoder transformer architecture which creates a summary with the decoding output. This information is then passed back out of the system, providing discrete information about the document and then prompting for a summarization which is generated as mentioned. A few options are also shown in order to show not just the closest but the cluster that are nearest to give more options.

#### FAISS

Similarly, for this model we imported data from [REDACTED] and then embedded, tokenized it to ensure a dense representation. Again, most of the embedding was done on the metadata rather than the text of the file itself as we are looking for topline similarities. For the next step, we indexed all of the data using the Facebook AI Similarity Search (FAISS) which created a model that we’d end up using for the retrieval aspect of the problem. For this model, we used a very similar query structure with a Name, Legal Abbreviation, Date, Jurisdiction, and custom option for people to look up topics and cases. This was to create more similarities between the two models which would allow for an ease of comparison.

Again, in a similar way, we prompted for a query in one of the categories which was also tokenized and embedded in a similar way to the metadata itself. With this representation, we passed it into the FAISS indexing which has its own methods of determining similarity and then printed out the cases which were closest to the prompt in addition to a brief snippet of the case itself for reference. This information is then passed back to the user and there are again multiple options shown to give access to other documents in the set.

### Discussion

### Conclusion

As we have shown, AI can be effectively used to expedite the research process of law work especially when it comes to citing proper sources and finding ones relevant to the current case. To return to the initial new story, our tool would really only help access sources which were not hallucinated by the large language model, the rest of the malpractice is not addressed but future works which combine retrieval augmented generation with other full texts of legal filing could fill in this gap. As machine learning techniques move forward, it's important to make sure that mechanics are put in place at each step to ensure quality of output and accuracy of the information. If not, AI companies may be in place to be sued themselves and the lawyers will be sure not to use ChatGPT for research.