Rishith Kyatham
112803780

CSE505 Project Report

CSE505 Project Part 1

1. The problem that I want to work on is a question answering system for legal documents. When I worked over my internship, I saw how the business analysts were struggling to extract relevant information efficiently as legal documents are long and tedious. They are wasting hours trying to find the information they are looking for when it can be done in seconds or to the minimum, narrowed down. My proposed solution would be developing a QA system that can answer queries based on legal documents. The **input** to this system would be legal documents passed in and the **output** would be the answer to the user queries including relevant passages from these legal documents of where it extracted this answer from. Now once a user retrieves these outputs, they can use it in their daily work to shorten searching time. The main **tool** I will be using to build this system is LangChain (source 8) which is a developer toolkit that helps develop LLMs from prototype to production. I would use the huggingFace (source 6) website for searching for good datasets and models. Kaggle (source 9) also has nice datasets so this isn't out of the option too. For evaluation of my results, I would base it on mostly recall, precision, recall, and F1 score metrics. I would also let a bunch of users try my project out such as my business major friends and ask them if the results they get are appropriate to what they are looking for. I would compare the solutions they get manually with my applications solution. Some **example uses** for this application are that it can be used for analyzing trends for researchers, extracting information for legal professionals, and for the normal user seeking legal information. Another constraint that it would solve is that usually in gpt, you can only upload a certain size of document into it. Now what if we want to input a whole book and ask it to do question answering tailored to the book only. This is another kind of use case where this application would flourish.

2. Since AI has taken a more precedence in today's world, there have been many applications like chatGpt that can answer questions from documents etc. However, there are restrictions and constraints to the extent of the usability of these applications. My application that I will be building will be more tailored for legal documents specifically. Some examples of general QA systems that have already been implemented are systems such as BERT, RoBERTa, and BERT (source 7) which have very good performances on various benchmarks. When I was searching through other open source applications, I saw ContraxSuite by LexPredict (source 1) which uses machine learning techniques to do this analysis on documents and find answers. Another approach I saw while looking into natural language processing and stanford content is the Jurafsky approach (source 2). This is a knowledge graph-based approach that extracts legal knowledge using natural language processing techniques to parse legal texts and extract relevant entities. It performs named entity recognition, relationship extraction, entity disambiguation, and schema mapping to solve problems. After researching through many processes, I think using LangChain (source 8) which contains various NLP functionalities is a good way. The code written through that is very readable and the solutions are going to depend

on the constraints set, LLMs used, the amount of training and testing done, hyperparameter tuning, evaluation, etc. I will aim to leverage these training language models that are already state-of-the-art such as BERT and RoBERTa (source 7) and make my project. When I was doing research, I was also looking into algorithms for performing similarity search since this will be used in my project and FAISS (source 3) is a C++ wrapper for python that is very efficient.

3. The tasks that I have to do will be listed below in bullet points in order :

- **Collecting data, finding the right datasets** : I would be gathering a diverse dataset of legal documents from various domains. I would get some datasets from kaggle, government websites, etc. When finding the right datasets, I will also check the credibility of the sources I get them from and the documents themselves. (Source 4) (Source 9) (Source 6)
- **Preprocessing the data** : I would use preprocessing techniques and tools developed in the NLP community to do so. Pandas would be a good tool for formatting the data. Text preprocessing libraries such as NLTK or spaCy are good tools for tokenizing texts and cleaning data.
- **Knowledge Representation** : Creating a structured schema design to represent legal concepts, relationships, and entities from legal documents. Capture the semantic connections between words and text. With the proper structure for taking in these documents, this facilitates efficient information retrieval and understanding within our QA system.
- **Question Answering** : Using advancements in natural language understanding and information from the NLP research community to improve our methods. I would use techniques from preexisting QA systems such as OpenAi's GPT models or Google's BERT (source 7) for understanding user queries better.
- **Information Retriever** : To efficiently retrieve from legal documents, I would use algorithms like Elasticsearch (source 10) or Apache Solr for efficient retrieval. I have to research more on what to use specifically, but this would be the general approach here. ChromaDB (source 14) is also an open-source vector storage system that is designed for storing and receiving vector embeddings so this could be a good tool to use while implementing this part.
- **Answer Generation** : Look into research in text generation and summarization. Example : Use transformer-based models like T5 (source 11) for generating accurate answers
- **Evaluation** : Use F1, recall, and precision to test how accurate the answers are. Do user inquiries by asking multiple users to test the answers and compare them manually. To check the speed, we could use libraries such as Time internally and check locally how long it takes to run depending on the constraints or models used.
- **HyperParameter Tuning** : Finding the optimal parameters to improve the model's performance so loop this back with evaluation. Use a validation set and perform cross-validation techniques (source 12) to evaluate the performance of different hyperparameters.
- **Deployment** : Make a UI that will look elegant and connect this with our project so the user could input queries in a box. I looked into streamlit (source 13) which might be a

feasible option for this. For interactive use, I would deploy this as a web application or API.

4. I would just use my evaluation step for evaluating my implementation. I would use precision, recall, and F1 score metrics to evaluate this system as I go. I would compare my results with QA systems like ContraxSuite or BERT, the ones I mentioned above and see how many iterations it would take for me to get close to the results.

Basic weekly plan (Approximate could be adjusted based on what's needed) :
Week 1 - 2 : Data collection and preprocessing and Knowledge Representation
Weeks 3 - 4 : Question Answering, Information Retriever, and Answer Generation
Weeks 5 : Evaluation and HyperParameter Tuning
Week 6 - 7 : Repeating the steps till I find a good enough model, constraints, answers, and evaluation metrics
Week 7 : Deploy after error checking for everything, make sure the task is accomplished

CSE505 Project Part 5 :

References used :

Source 1 : https://github.com/LexPredict/lexpredict-contraxsuite (LexPredict ContraxSuite)
Source 2 : https://web.stanford.edu/~jurafsky/slp3/14.pdf (Question Answering Research for jurafsky)
Source 3 : https://github.com/facebookresearch/faiss (a library for efficient similarity search)
Source 4 : https://case.law/ (For tailored datasets)
Source 5 : https://www.nltk.org/ (For Natural Language Toolkit)
Source 6 : https://huggingface.co/ (Where to find datasets and models to test)
Source 7 :
https://www.nvidia.com/en-us/glossary/bert/#:~:text=BERT%20is%20a%20model%20for,RoBERTa%2C%20ALBERT%2C%20and%20DistilBERT. (Research on BERT)
Source 8 : https://github.com/langchain-ai/langchain (For Lang Chain)
Source 9 : https://www.kaggle.com/ (For finding good datasets on Kaggle)
Source 10 : https://www.knowi.com/blog/what-is-elastic-search/ (reading up on elastic search)
Source 11 : https://medium.com/analytics-vidhya/t5-a-detailed-explanation-a0ac9bc53e51 Explanation on T5
Source 12 :
https://www.turing.com/kb/different-types-of-cross-validations-in-machine-learning-and-their-explanations Cross validation techniques
Source 13 : https://streamlit.io/ Stream lit (I might use for the UI portion for my application)
There were some other implementations and sources I found but those cost money for demos.
Source 14 : https://www.trychroma.com/ (vector storage system)

Work Cited

Chen, Qiurui. “T5: A Detailed Explanation.” *Medium*, Analytics Vidhya, 11 June 2020, medium.com/analytics-vidhya/t5-a-detailed-explanation-a0ac9bc53e51.

“Different Types of Cross-Validations in Machine Learning.” *Different Types of Cross-Validations in Machine Learning.*, Turing Enterprises Inc, 11 Mar. 2022, www.turing.com/kb/different-types-of-cross-validations-in-machine-learning-and-their-explanations.

“Elasticsearch: What It Is, How It Works, and It’s Usage.” *Knowi*, 28 Feb. 2024, www.knowi.com/blog/what-is-elastic-search/.

Facebookresearch. “Facebookresearch/Faiss: A Library for Efficient Similarity Search and Clustering of Dense Vectors.” *GitHub*, github.com/facebookresearch/faiss. Accessed 15 Mar. 2024.

“Hugging Face – the AI Community Building the Future.” *Hugging Face* –, huggingface.co/. Accessed 15 Mar. 2024.

Langchain-Ai. “Langchain-Ai/Langchain: 🦜🔗 Build Context-Aware Reasoning Applications.” *GitHub*, github.com/langchain-ai/langchain. Accessed 15 Mar. 2024.

LexPredict. “LexPredict/Lexpredict-Contraxsuite: LexPredict ContraxSuite.” *GitHub*, github.com/LexPredict/lexpredict-contraxsuite. Accessed 15 Mar. 2024.

NLTK, www.nltk.org/. Accessed 15 Mar. 2024.

Question Answering and In- Formation Retrieval, web.stanford.edu/~jurafsky/slp3/14.pdf. Accessed 16 Mar. 2024.

Streamlit • A Faster Way to Build and Share Data Apps, streamlit.io/. Accessed 16 Mar. 2024.

US Laws, Cases, Codes, and Statutes | Findlaw Caselaw, caselaw.findlaw.com/. Accessed 16 Mar. 2024.

“What Is Bert?” *NVIDIA Data Science Glossary*, www.nvidia.com/en-us/glossary/bert/#:~:text=BERT%20is%20a%20model%20for,RoBERTa%2C%20ALBERT%2C%20and%20DistilBERT. Accessed 15 Mar. 2024.

“Your Machine Learning and Data Science Community.” *Kaggle*, www.kaggle.com/. Accessed 15 Mar. 2024.

“The AI-Native Open-Source Embedding Database.” *Chroma*, www.trychroma.com/. Accessed 15 Mar. 2024.

CSE505 Project Part 2 :

Design document : I attached the python file of the design document to the zip

Brief summary of overall design : This design system features an API and driver program for a Legal Document Question Answering (QA) System using LangChain. The API handles document uploading and query answering, while the driver manages workflow and system evaluation, both utilizing LangChain and ChromaDB for language modeling and database functions. The input consists of legal documents for uploading and user queries for answering, while the output includes status messages for document uploading and answers with relevant reference documents for user queries.