FA2023 CS410 Final Project Documentation
Using BM25 and Retrieval Augmented Generation to Help CS441 Students

Hannah Benig (hhlim2@illinois.edu)

1. Introduction

Imagine you are a student working on a study guide for an exam using information from a PDF version of your textbook. When looking for answers, you have two options: try searching for a keyword in the PDF that might be relevant to your question or read through a whole section where the answer might be. You could also try searching your question in Google, but then you might receive results that are less relevant or provide incorrect information within the scope of your class. In this project, a Streamlit user interface has been developed to assist students by allowing them to ask a natural language question as they would in Google and receive accurate and relevant responses with information directly from the textbook.

To achieve success, this project utilizes the BM25 scoring function, a retrieval augmented generation (RAG) workflow, and a large language model (LLM). It allows users to ask direct questions about the content in the CS441 textbook and quickly find an answer. The user interacts with a Streamlit UI to enter their question and receive an output consisting of an answer, the pdf pages where the answer was found, and the average BM25 score between the query and the relevant passages.

Three tools were developed for this project

1. A Streamlit User Interface
2. A BM25 scoring class.
3. A RAG workflow.

2. Project Architecture & Tools

The project architecture is shown in Figure 1.

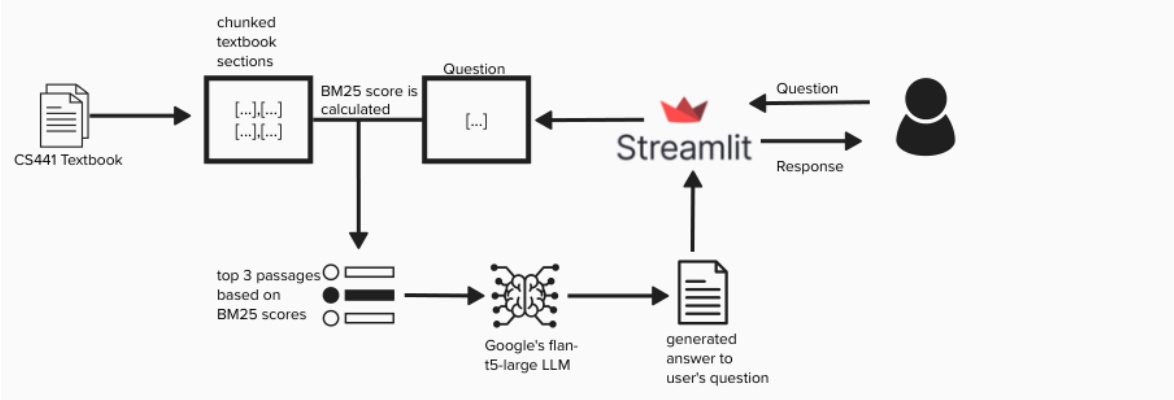

Figure 1: Project Architecture

When the Streamlit web app starts up, the CS441 textbook is loaded into the application using Langchain's PyPDFLoader and serves as our document language model. This is used in combination with Google's Flan-t5-Large LLM, which serves as our background language model. The PDF is then chunked into smaller sections and initialized as the BM25 corpus. When a question is asked in the Streamlit UI, we will use BM25 scoring to find the 3 sections containing information most similar to our question. We will then pass our question and the relevant information to the LLM so that our answer best reflects the information presented in the textbook. The generated response is returned to the user in the Streamlit app, and the PDF preview is updated to the relevant page.

2.1 Streamlit User Interface

The user interacts with a simple Streamlit UI to ask questions, receive relevant responses, and navigate the textbook. Streamlit is a Python library that allows developers to easily create a web application that can be deployed locally. The Streamlit UI is shown in Figure 2.

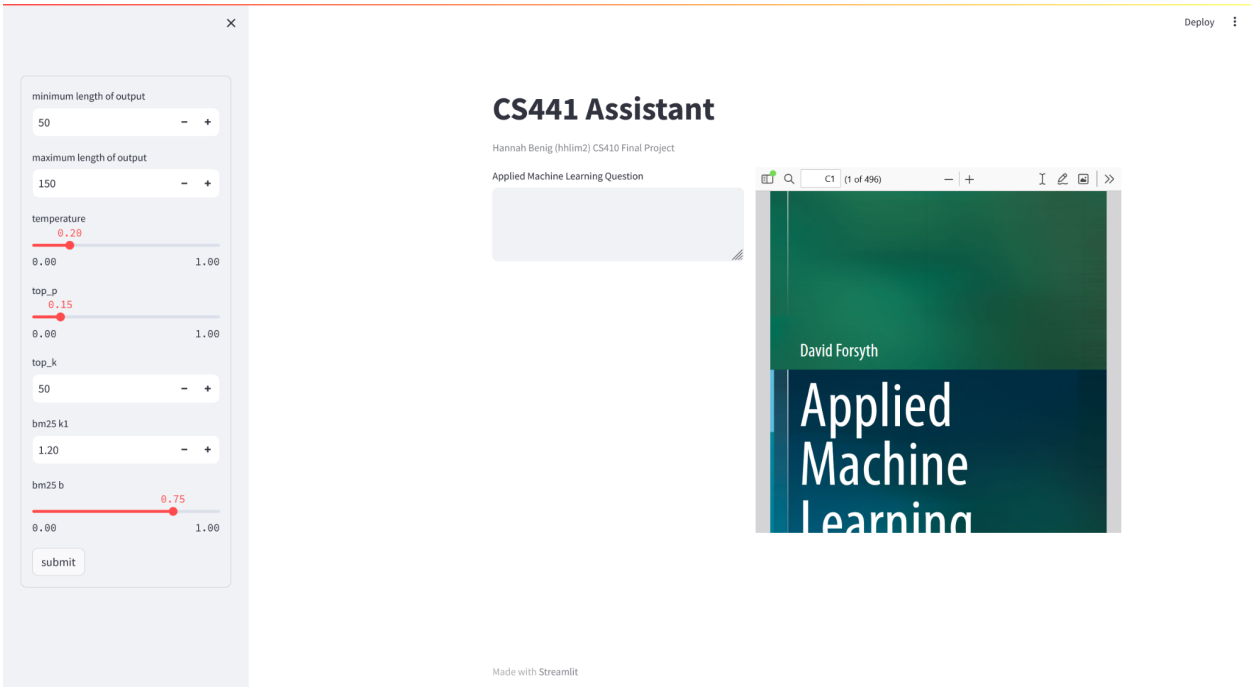

Figure 2: Streamlit UI

2.1.1 Highlighted Streamlit Functions

1. **Session State.** Streamlit allows us to set session state variables, removing the need to reload the whole Python script every time a question is asked or a model parameter is changed. We know that variables such as the corpus document (the CS441 textbook) and the LLM will not change during our Q&A session, so there is no need to reload them

each time a question is asked, which will happen if the session state variables are not initialized.

2. **Forms.** The parameters for the LLM and BM25 function can be set in the left-hand sidebar of the UI. These parameters are stored as variables within a Streamlit Form. If a value changes, the application waits until the submit button is clicked before using the new parameters. Using this form removes unnecessary load times so that if multiple parameters are changed, they are loaded all at once rather than one at a time.
3. **PDF Display.** A PDF of the CS441 textbook is displayed in the web app for the user to navigate through. When a question is asked, the display dynamically updates to show the page where the passage with the highest BM25 score can be found, allowing the user to see the context of the answer to their question.

2.2 BM25 Scoring Class

BM25 is a ranking function that determines the most relevant documents to the user's query based on the number of times that the words in the query appear in the document.

When the web app starts up, the textbook is chunked into smaller sections and initiated as the BM25 corpus; the length of each chunk and the average length of all chunks are set as variables, and stop words are removed. The BM25 score is calculated for each textbook chunk and used to determine which passages are most similar to the user's question. This provides the LLM with the information needed to answer the question.

For each word in the question entered by the user, both the document frequency and inverse document frequency of the word are calculated. The k1 and b parameters are initialized to the default parameters of 1.2 and 0.75, respectively, but can be changed by the user in the Streamlit UI. A BM25 score is then calculated for each section using the following equation:

$$score(q, D) = \sum_{i}^{n} IDF(q_i) \frac{f(q_i, D) * (k1 + 1)}{f(q_i, D) + k1*(1-b + b*\frac{dl}{avgdl})}$$

The three textbook sections with the highest BM25 scores are extracted from the corpus and fed to the LLM in the prompt as context to the question.

2.3 Retrieval Augmented Generation (RAG)

Large language models are trained on a large corpus of information, but if your question falls in a category not included in the training information, you may not receive a valid answer. Retrieval Augmented Generation (RAG) helps us to avoid this problem. In the RAG workflow, the 3 passages with the highest BM25 scores are concatenated and provided in the model prompt along with the user query. By using this workflow, we ensure that a more accurate and relevant answer is generated in response to the user question because we are providing the model with specific information where the answer can be found.

2.3.1 RAG Workflow Considerations

1. **The Dataset.** In the project proposal, the original idea was to use the *NewsCatcherAPI* to retrieve news articles that were relevant to the user's query and use the articles as the background language model from which we could draw context that would be passed to the LLM. After much experimentation, it was found that the *NewsCatcherAPI* was limited in its retrieval abilities and struggled with multi-word searches. The *NewsCatcherAPI* program also required an API key that would expire at the end of the semester or require additional payment for continued use. With these issues in mind, the dataset was changed to the CS441 textbook as it is available to students through the UIUC library and contains many topics that overlap with CS410. Similarly, we use the CS441 textbook as the background language model and draw context related to the user question to aid the LLM in answering the query. Additionally, users are able to swap out this PDF for another or even a whole directory of files if they would like to use the RAG workflow for another topic.
2. **The Model.** Many models were experimented with during the development of this project. Models tested include roBERTa, Flan-UL2, Llama2 (7B and 70B), Google's MPT, GPT2, and Flan-t5 (XS, S, Base, L, and XL). While models like Flan-UL2 and Llama2 have been shown to perform very well with generative tasks like Q&A and RAG, they were too large to run locally. The roBERTa, MPT, and GPT2 were unable to generate any kind of reasonable answer. The Flan-t5 models worked the best. The Large model performed better than XS, S, and Base, which makes sense given the increased size of the training dataset and was able to run locally, unlike the XL model. If desired, users are able to swap out the Flan-t5-Large model for any Huggingface model of their choosing.
3. **The Hyperparameters.** Different combinations of model hyperparameters were experimented with. The parameters that yielded the best results are set as the default values in the left sidebar of the Streamlit UI. Parameters tested include min_length, max_length, temperature, top_p, and top_k. Parameters like min_length and max_length refer to the length of the response and can be tricky to set because if too low, the response may not contain all the information to answer the question, and if too high, the response may be repetitive. Generally, for parameters like temperature, top_p, and top_k, the higher the value, the more variance in the answer. Users are able to change the parameters to see how they affect the response.
4. **The Prompt.** Multiple prompts were experimented with to see which would yield the best results. Prompts included:

```
1. prompt1 = f"Use the following context: {relevant_passages} to answer
   the question: {input_query}."
2. prompt2 = f"Context: {relevant_passages}, Question: {input_query},
   Answer: "
3. prompt3 = f"{relevant_passages} \n\n {input_query}"
```

It was found that prompt3 worked the best, providing only the relevant passages and the user query. Adding additional instructions did not improve the answer and, in some instances, seemed to confuse the model.

3. Evaluation

When the project was proposed, success was to be determined by whether or not the application could correctly answer a list of questions with greater than 75% accuracy. The application was tested with various question types and found that the system performs best with "what is…" questions. Models like Flan-UL2 and Llama2 would be better suited for this type of application as they are able to better more nuanced questions like "explain…" or "compare/contrast…". In this project, we are limited by models available on Huggingface and their size so that we can load the model locally rather than paying to use a cloud-hosted model. The evaluation results are shown in Figure 3.

	Question	RAG Output	Valid (y/n)	Non-RAG Output	Valid (y/n)
2	what is agglomerative clustering?	Agglomerative clustering is a clustering algorithm that starts with each data item being a cluster, and then merges clusters recursively to yield a good clustering. In divisive clustering, you start with the entire dataset being a cluster, and then split clusters recursively to yield a good clustering.	y	Agglomerative clustering is a clustering algorithm that starts with each data item being a cluster, and then merges clusters recursively to yield a good clustering. In divisive clustering, you start with the entire dataset being a cluster, and then split clusters recursively to yield a good clustering.	y
3	what is a support vector machine?	SVM is a linear classifier trained with the hinge loss. The hinge loss is a cost function that evaluates errors made by two-class classifiers. If an example is classified with the right sign and a large magnitude, the loss is zero; if the magnitude is small, the loss is larger; and if the example has the wrong sign, the loss is larger still. When the loss iszero, it grows linearly in the magnitude of the prediction.	y	<pad> A support vector machine (SVM) is a machine learning algorithm that learns to predict the likelihood of a given input. It is a generalization of the LSVM, which is a support vector machine. A SVM is a machine learning algorithm that learns to predict the likelihood of a given input.</s>	y
4	what is the difference between agglomerative and divisive clustering?	In agglomerative clustering, you start with each data item being a cluster, and then merge clusters recursively to yield a good clustering (Procedure 8.1). The difficulty here is that we need to know a good way to measure the distance between clusters, which can be somewhat harder than the distance between points. In divisive clustering, you start with the entire dataset being a cluster, and then split clusters recursively to yield a good clustering (Procedure 8.2).	y	<pad> Agglomerative clustering is a method of clustering that uses a recursive algorithm to find the most common clusters in a given population. It is a generalization of the divisive clustering method.</s>'	n
5	explain how object detection works	Object detectors work by passing image boxes that are likely to contain objects into a classifier. The classifier gives scores foreach possible object in the box. Multiple detections of the same object by overlapping boxes can be dealt with by non-maximum suppression, where higher-scoring boxes eliminate lower-scoring but overlapping boxes. Boxesare then adjusted with a bounding box regression step. 18.2.2 Selective Search The simplest procedure for building boxes is to slide a window over the image. This is simple, but works rather badly. It ignores important image evidence. Objects tend to have quite clear boundaries in images. For example, if you are looking at	y	<pad> a laser is used to detect objects that are moving. a laser is used to detect objects that are moving. a laser is used to detect objects that are moving. a laser is used to detect objects that are moving. a laser is used to detect objects that are moving.</s>'	n
6	what dataset should be used for object detection?	,18.1. Image Classification 426 PASCAL VOC 2007 ImageNet ImageNet Large Scale Visual Recognition Challenge (ILSVRC2012) ImageNet Large Scale Visual Recognition Challenge (ILSVRC2012) ImageNet Large Scale Visual Recognition Challenge (ILSVRC2012)	y - 1/2 credit since it just lists datasets rather than answering the question	<pad> a large number of objects in a room. a large number of objects in a room. a large number of objects in a room. a large number of objects in a room. a large number of objects in a room.</s>'	n
7	what is linear regression	A linear regression takes the feature vector xand predicts xT,f o r some vector of coefficients . The coefficients are adjusted, using data, to produce the best predictions.	y	linear regression is a type of regression that shows the relationship between two variables.'	n
8					
9		Score	5.5/6		2/6

Figure 3: Evaluation Results

The results showed that our RAG application could answer 5.5/6 or 91.76% of the questions correctly. The results in Figure 3 show that we met the desired 75% success rate.

During the evaluation process, we added an additional metric and compared the results of the developed application to the output of the Flan-t5-Large model with no context given in the prompt. We saw that when asking the Flan-t5-Large model questions with no context, only 2/6 or 33% of the questions were answered correctly. Some of the questions asked resulted in repetitive, nonsense answers unrelated to the question.

By utilizing a RAG workflow, we improved an off-the-shelf open-source model to provide users with better answers to questions about topics covered in the CS441 textbook. Additionally, we

provided users with a reusable workflow that can be adapted for any corpus of information they choose. Overall, the goals set for this project were accomplished and we determine this project to be a success.

4. Installation and Usage

Instructions on how to install and run the web app locally, along with example questions that can be used for testing. Two methods of running the application are provided.

1. Locally:
    - Recommended specs: 32GB RAM, i7-12700H Processor
    - A computer with worse specs might run very slowly or not at all
2. Google Colab:
    - Will run slowly but does not depend on personal computer specs

4.1 Prerequisites for Local

- Mac OS, Linux, or WSL (on a Windows PC)
    - WSL installation: https://learn.microsoft.com/en-us/windows/wsl/install
        - Type 'wsl' in terminal to activate
- Python 3.11

4.2 Setup and Run Streamlit App in Google Colab

The Colab notebook contains the same code from the main.py and bm25.py files but allows you to utilize the T4 GPU runtime.

Figure 4: Google Colab Notebook Setup

Instructions

1. Download the CS441 Textbook.pdf file from https://github.com/hhlim2/CourseProject-HB/blob/main/CS441%20Textbook.pdf
2. Open Google Colab Link in **Chrome** (make a copy and run in your personal Google account): https://colab.research.google.com/drive/1lxnP1LTVJ0fClyiLbsI97CLCa1EieNg4?usp=sharing
3. Upload the CS441 Textbook.pdf file into the Google Colab Notebook Files folder (numbers 1 & 2 in Figure 4)
4. Connect to the T4 GPU runtime in the Google Colab Notebook (number 3 in Figure 4)
5. Run all cells

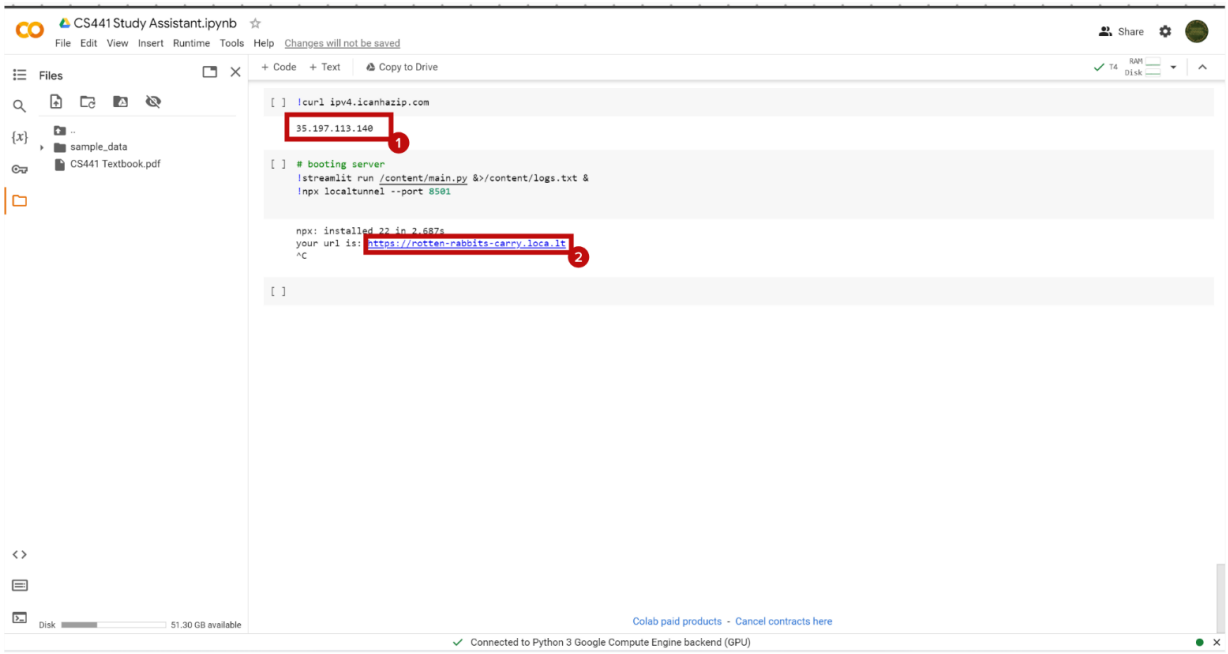

Figure 5: IP Address and Localtunnel Link

6. Copy the IP Address contained in the output of cell 3 (number 1 in Figure 5)
7. Click on the link in the output of cell 4 (number 2 in Figure 5)

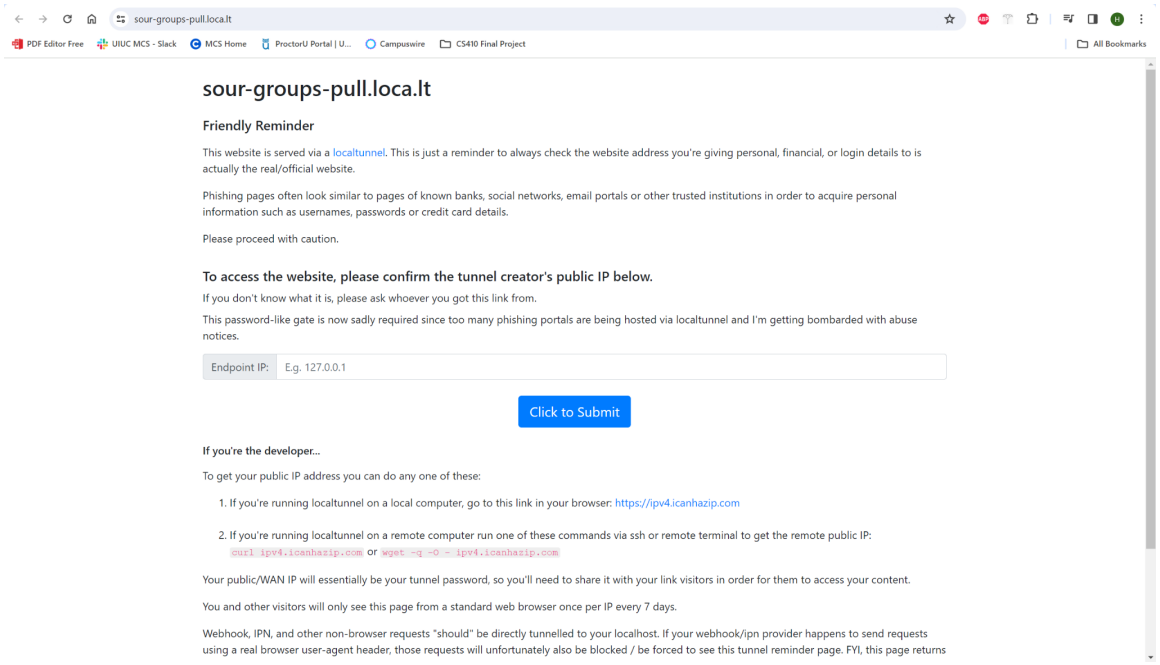

Figure 6: Example Localtunnel Webpage

8. The link from the previous step will bring you to a page similar to the one in Figure 6. Enter the IP Address you copied in step 6 in the “Endpoint IP” box and click Submit.
9. You should be brought to the Streamlit App shown in Figure 2.

4.3 Setup and Run Streamlit App Locally

1. Clone the project repository

```
git clone https://github.com/hhlim2/CourseProject-HB.git
```

2. Create a virtual environment within the repository directory

```
cd CourseProject-HB
python3 -m venv env
```

3. Activate the virtual environment

```
source env/bin/activate
```

4. Upgrade pip and install all dependencies

```
pip install --upgrade pip
pip install -r requrements.txt
```

5. Run the application

```
streamlit run main.py
```

Open the localhost link in **Firefox** for full features (PDF display does not work in Chrome)

4.4 Usage

1. Input your question regarding topics covered in the CS441 textbook to receive an answer. The PDF display will bring you to the page/section where the answer can be found.
2. You can change the hyperparameters in the left sidebar to see how changing parameters affects the answer. The default parameters have been set to values where the best response is generated.

4.4.1 Example Questions

Please note that LLMs are probabilistic models which can lead to variations in the returned response. Unfortunately, Huggingface models do not have a random_seed parameter to ensure that the same response is always returned. Additionally, small changes, such as adding different punctuation or changing words, can lead to drastically different answers. Example questions are provided below to ensure the application can be tested successfully.

1. what is agglomerative clustering
2. explain how object detection works
3. what is linear regression