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

Hannah Benig (hhlim2@illinois.edu)

1. Introduction

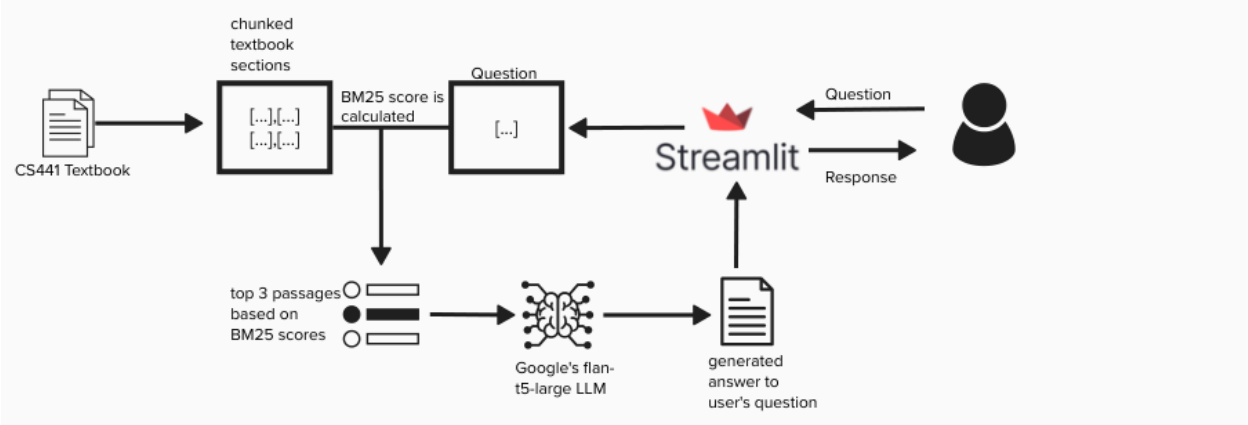
When searching through a PDF, users are unable to ask a direct question about the content; instead, they must search a keyword and look at each result to determine if it is relevant to their query or not. When searching in Google, the user is able to ask a direct question but may receive results that are not relevant or incorrect within the scope of their class. This project utilizes BM25, retrieval augmented generation (RAG), and a large language model (LLM) to address both problems by allowing users to ask direct questions of content in the CS441 textbook and quickly find an answer. The output will be a direct answer to the question, the pdf pages where the answer was found, and the average BM25 score between the query and the relevant passages. The user enters their question and receives a response in the Streamlit UI while the RAG workflow, BM25 scoring function, and LLM run in a Python script.

Three tools were developed for this project

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

2. Project Architecture & Tools

The project architecture is shown in the following figure:



When the web app starts up, the CS441 textbook is loaded into the application using Langchain's PyPDFLoader and serves as our document language model that is used in combination with Google's Flan-t5-Large LLM. The PDF is then chunked into smaller sections

that we will pass as context to our model. Next, we initialize the textbook chunks as our BM25 corpus using the BM25 scoring class we've developed. When the user enters a question in the Streamlit front end, the BM25 score is calculated between each chunked section and the question. The 3 passages with the highest BM25 scores are returned and passed to the LLM along with the user's question. An answer is generated and returned to the user in the Streamlit app, and the PDF preview is updated to the relevant page.

2.1 Retrieval Augmented Generation (RAG)

Large language models are trained on a large corpus of information, but if your question falls into 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 by obtaining passages related to our question from a relevant dataset so that a more accurate answer is generated when the question is asked to the large language model.

2.1.1 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 context 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. 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.** The results of hyperparameter testing are shown in the left sidebar. Parameters tested include: min_length, max_length, temperature, top_p, and top_k. Parameters like min_length and max_length can be tricky because too low, the response may not contain all the information to answer the question, and 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.
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; adding additional instructions did not improve the answer and, in some instances, seemed to confuse the model. The model did not need much additional information beyond the context information and the user query.

2.2 BM25 Scoring Class

BM25 is a ranking function that calculates the most relevant documents based on the user's query.

When the web app starts up, the textbook chunks created in an earlier step are 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. For each word in the query, 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. The BM25 score is then calculated 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 Streamlit User Interface

Streamlit is a Python library that provides in-depth and simple instructions to create a user interface that can be deployed locally.

2.3.1 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 model and BM25 parameters found in the left-hand sidebar are located within a Streamlit Form. If a value changes in this form, the application waits until the submit button is clicked before setting all the values. Using this form removes unnecessary load times so that if multiple parameters are changed, they are set all at once rather than one at a time.
3. **PDF Display.** The PDF display allows the user to quickly find the page or section where the answer to their question is found. Streamlit allows us to embed a PDF display directly into the page and dynamically update it when a new question is asked.

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. During the evaluation process, this was taken one step further, and the results of the developed application were compared to the output of the Flan-t5-Large model with no context given in the prompt. The application was tested with various question types, including what is…, explain…, and compare/contrast. The evaluation results are shown in the following figure:

	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	how do we know if a matrix is symmetric	if the x-axis is symmetric and the y-axis is symmetric, then the matrix is symmetric. if the x-axis is symmetric and the y-axis is symmetric, then the matrix is not symmetric.	y	<pad> If the matrix is symmetric, the vertices are parallel. If the vertices are not parallel, the vertices are perpendicular. If the vertices are not parallel, the vertices are perpendicular.</s>'	n
8					
9		Score	5.5/6		2/6

The results showed that our RAG application could answer 5.5/6 or 91.76% of the questions correctly. The 1/2 point deduction was from the question, “What dataset should be used for object detection?”. The response was a list of datasets and object recognition challenges rather than just datasets. In comparison, when asking the Flan-t5-Large model questions with no context, only 2/6 or 33% of questions were answered correctly. Half of the questions asked resulted in repetitive, nonsense answers unrelated to the question.

Based on these results, we have determined that our approach was successful.

4. Usage

4.1 Prerequisites

- 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
    - wsl can be quite slow; another option is to remote into a Linux machine via FastX from UIUC EWS: https://answers.uillinois.edu/illinois.engineering/page.php?id=81693
- Python

4.2 Setup and Run Streamlit App

1. Clone the project repository

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

2. Create a virtual environment

```
python3 -m venv env
```

3. Activate the virtual environment

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

4. Install all dependencies

```
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)

4.3 Usage

1. Input your question regarding topics covered in the cs441 textbook and receive an answer. The PDF viewer will bring you to the page/section where the answer can be found.
2. You can change the hyperparameters in the left window to see how tweaking parameters changes the answer. The default parameters have been set to values where the best response is generated.

4.3.1 Example Questions

1. What is agglomerative clustering?
2. What is the difference between agglomerative and divisive clustering?
3. Explain how object detection works.