Team Project Final Report

Project Title: Generative Search

Team Members:
- Team Captain: Mohammad Tamim (NetID: tamim2)
- Team Members:
  - Timothy Cannella (NetID: tdc5)
  - Luke Freitag (NetID: lukegf2)

Introduction

Our Generative Search project represents a convergence of cutting-edge technologies in information retrieval, natural language processing, and machine learning. In response to the limitations of traditional search engines, our team, has embarked on a mission to develop a sophisticated Generative Search system. This system leverages the power of Retrieval Augmented Generation (RAG) combined with Large Language Models (LLMs) to provide users with contextually accurate and informative responses to queries related to the "CS 410 Text Information Systems" course.

This documentation serves as a comprehensive guide to understanding the intricacies of our Generative Search system. From the initial planning stages to the development, testing and evaluation phases, we will delve into the methodologies, tools, and technologies employed. The documentation is structured to cater to both users seeking insights into the system's functionality and developers interested in extending or improving upon our codebase.

In the upcoming sections, we'll take a deep dive into a comprehensive journey through our Generative Search system. We'll start with a look at our meticulous project planning, followed by an exploration of the system's design, the intricacies of code development, crystal-clear instructions on installation and usage, and a thorough examination during the evaluation process. Our narrative will culminate by spotlighting the collaborative synergy within our team, emphasizing the distinctive contributions of each member. Our commitment to transparency and clarity is foundational, assuring that the evaluation of our project authentically mirrors the dedication and expertise invested by every team member.

Planning

Our approach to developing the Generative Search system was organized into four clear tracks, ensuring a streamlined and efficient process:

1. **Track 1: Data Collection**
   a. Transcript Collection: In this stage, our focus was on gathering all transcripts for "CS 410 Text Information Systems" from Coursera. The objective was to organize this textual data, including lecture headings, and save it in a structured text file.

   b. Q&A Dataset Creation: We utilized ChatGPT to generate questions and answers for each lecture, aiming to build a versatile dataset. The goal was to produce 10

paraphrased versions for every question while ensuring consistent answers. For instance, if the original question was " Why is it important to process text data?", we generated 10 other ways to ask it. The result is a .csv file containing "question" and "response" columns. This dataset enhances our system's ability to identify similarity during document retrieval in the Generative Search system.

2. **Track 2: Design, Development, and Testing**
    a. Designing the Generative Search System: Our initial step involved crafting the design for the Generative Search system. We created a visual workflow to outline how the system would function.

    b. Developing the Generative Search System:
        i. *Convert Questions to Embedded Vectors:* Using Google Colab and Python, we developed code to transform questions from the Q&A dataset into context-embedded vectors. The results were stored in a .csv file with three columns: "question," "response," and "embedded_questions."

        ii. *Online Generative Search Module:* This critical module, utilizing RAG, was developed in Google Colab. It accepted user input, preprocessed it by converting it into an embedded vector, performed a search to find the most similar question within 'embedded_questions' using Cosine similarity metric. Once identified, the system retrieved the corresponding response, utilizing LLM along with prompt engineering techniques to generate an answer based on the gathered context.

    c. Testing: Rigorous testing was a crucial aspect of our development process. We conducted thorough testing to ensure the functionality and accuracy of the Generative Search system. This phase ensured that the system operated as intended, providing reliable results.

3. **Track 3: Results Evaluation**
    a. Evaluation Sample Set Creation: Initially, we created a set of 50 questions for evaluation. 25 questions were identical to those in the Q&A dataset, focusing on assessing false negatives. The remaining 25 questions were unrelated, aimed at evaluating false positives. The outcome was a .csv file with three columns: "question," "response," and "generated_response." For the first 25 questions, accurate responses were obtained from the Q&A dataset, while the accurate response for the unrelated questions was uniformly set as: "I'm sorry, but I don't have expertise in this topic; my training data is limited to only 'CS 410 Text Information Systems.'"

    b. Quantitative Evaluation: Using Python, a cosine similarity analysis was conducted between the "response" and "generated_response" for the initial 25 questions related to the course. Cosine similarity scores, reported with three decimal points (e.g., 0.789), deemed responses with a score above 0.5 as accurate. For the second set of 25 unrelated questions, a score of 1 was assigned if the "generated_response" matched the predefined response, and 0 otherwise. The results were saved in a .csv

format, featuring four columns: "question," "response," "generated_response," and "cosine similarity."

c. Task 3: Qualitative Evaluation: This phase involved a manual review of the 50 questions and responses by team members. Accuracy and relevance of the "generated_response" were assessed in comparison to the "response." A .csv file was created, encompassing four columns: "question," "response," "generated_response," and "is_generated_response_relevant?" The "is_generated_response_relevant?" column contained values of 0 or 1, indicating the relevance of the generated response.

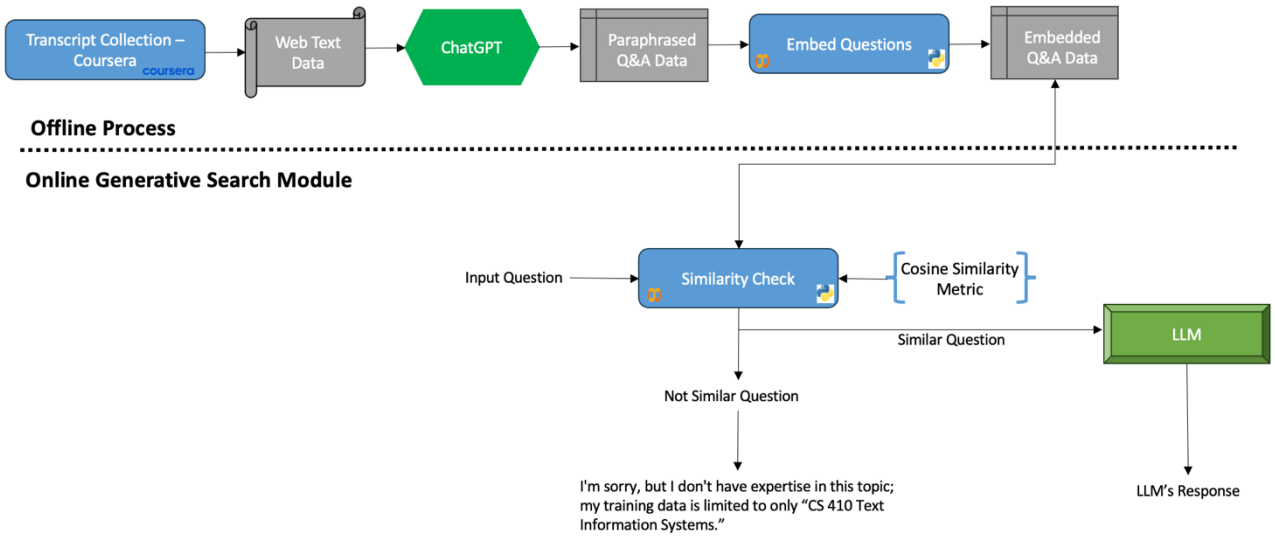
4. **Track 4: Reporting**
   a. Prepare the Progress Report: We crafted an initial progress report for submission, outlining the status of our project, highlighting achievements, and indicating the direction we were heading.

   b. Create the Final Report: A comprehensive final report was developed as part of this submission, encompassing project details, the methodology employed, a summary of results, evaluation outcomes, and key recommendations.

   c. Prepare the Presentation: In addition to the written reports, we successfully prepared a presentation to effectively communicate our project's goals, progress, findings, and recommendations.

System Design

In this section, we elaborate on the design of our Generative Search system, which is presented below.

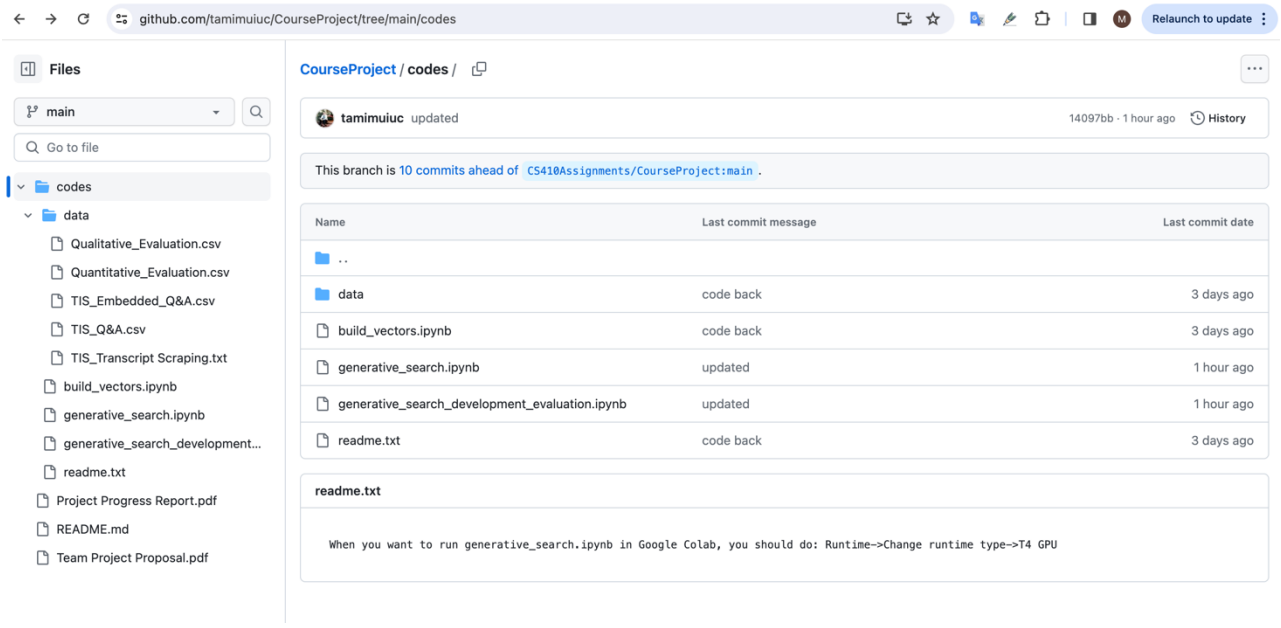


The system development is split into two processes: the offline process and the Online Generative Search module. In the offline process, we initiated manual transcript collection from Coursera, storing this textual data into a text file. Subsequently, leveraging ChatGPT, we generated a question and response dataset from the text file. For each question, we systematically generated 10 paraphrased versions, enriching the dataset. The next step involved embedding questions into vectors using the Bert sentence transformer (the all-MiniLM-L6-v2 model). These embedded paraphrased questions and responses were then stored in a.csv file.

Moving to the Online Generative Search module, this component serves as the user interface for posing questions related to "CS 410 Text Information Systems". When a user enters a question, the system checks for semantic similarity with a cosine threshold > 0.7 to the embedded questions. If a match is found, the system retrieves the response with the highest cosine score. This response, along with the user's question, is then fed into the Dolly-v2-3b LLM to generate a contextualized response. In cases where the user's input question lacks semantic similarity (cosine <= 0.7) with the embedded questions, the system generates a default response: "I'm sorry, but I don't have expertise in this topic; my training data is limited to only 'CS 410 Text Information Systems.'" This encapsulates the entirety of our system design.

An overview of the function of the code

In the provided GitHub repository link, you'll find three essential Python notebooks for our project:



1. **build_vectors.ipynb:** This notebook reads the "TIS_Q&A.csv" file from the "data" folder, containing paraphrased question and answer sets. It utilizes the Bert sentence transformer, the all-MiniLM-L6-v2 model, to convert questions into embedding vectors. The resulting vectors are then stored in "TIS_Embedded_Q&A.csv", featuring three columns: "question," "response," and "embedded_questions."

2. **generative_search_development_evaluation.ipynb:** In this notebook, the development and evaluation of the Generative Search System take place. It reads

"TIS_Embedded_Q&A.csv", randomly selects 25 questions related to "CS 410 Text Information Systems", and another 25 unrelated questions for evaluation is generated (e.g., "What is Transmission Control Protocol?"). The generative_search() function is employed, checking semantic similarity with a cosine score threshold > 0.7 to embedded questions. If a match is found, the system retrieves the response with the highest cosine score and generates a contextualized response using the Dolly-v2-3b LLM. For questions lacking semantic similarity, a default response of "I'm sorry, but I don't have expertise in this topic; my training data is limited to only 'CS 410 Text Information Systems.'" is generated. Then, the notebook conducts quantitative evaluation by performing cosine similarity analysis and generating the input file for qualitative evaluation, outputting respective CSV files ("Quantitative_Evaluation.csv" and "Qualitative_Evaluation.csv").

3. **generative_search.ipynb:** This notebook encapsulates the main Generative Search System. It reads "TIS_Embedded_Q&A.csv", prompts the user for an input question, and generates a contextually related response if the question's cosine score exceeds 0.7. Otherwise, it prints "I'm sorry, but I don't have expertise in this topic; my training data is limited to only 'CS 410 Text Information Systems.'" This system serves as a search tool for "CS 410 Text Information Systems" course contents.

In summary, the provided code encompasses the creation of embedding vectors, development, and evaluation of the Generative Search System, and the standalone Generative Search module. These components collectively form a robust tool for exploring and retrieving information from the "CS 410 Text Information Systems" course contents.

Implementation Overview

Within the "generative_search.ipynb" notebook, our team has crafted an advanced Generative Search system, marking a transformative stride in information retrieval, natural language processing, and machine learning. This sophisticated system stands as a testament to ingenuity, tackling and surmounting the constraints of traditional search engines by seamlessly incorporating RAG and LLM. The result is an intricately detailed mechanism designed to deliver users contextually precise responses to queries specifically related to the "CS 410 Text Information Systems" course.

The "generative_search.ipynb" notebook begins by loading the Dolly-v2-3b model from HuggingFace and utilizes LangChain, incorporating a template for calling the Dolly-v2-3b model. Furthermore, it incorporates the Bert sentence transformer, specifically the all-MiniLM-L6-v2 model, for embedding purposes. Subsequently, the notebook reads the "TIS_Embedded_Q&A.csv" file to compare user questions with those related to the "CS 410 Text Information Systems" course.

Central to this notebook is the generative_search() function. When a user's question is provided, this function converts it into an embedding using the Bert Sentence transformer. It then calls the find_top_response() function, which checks if the user's question is semantically similar to embedded questions in "TIS_Embedded_Q&A.csv" with a cosine threshold above 0.7. If a match is found, the system retrieves the response with the highest cosine score. The user's question, along with this retrieved response, is then fed into the Dolly-v2-3b LLM using the llm_context_chain.predict() function to generate a contextualized response.

In cases where the user's input question lacks semantic similarity (cosine <= 0.7) with the embedded questions, the system generates a default response: "I'm sorry, but I don't have expertise in this topic; my training data is limited to only 'CS 410 Text Information Systems.'"

Running Generative Search System

To seamlessly harness the capabilities of the Generative Search System encapsulated in the "generative_search.ipynb" notebook, follow these step-by-step instructions for setup and execution:

1. **Preparation:** Save the "generative_search.ipynb" file in your Google Drive under "My Drive/CoLabNotebooks". Ensure the existence of the "data" folder in the same directory as the notebook, containing the file "TIS_Embedded_Q&A.csv".

2. **Google Colab Setup:** Open Google Colab. Navigate to "File" -> "Open notebook" -> "Google Drive". Locate and open the "generative_search.ipynb" notebook.

3. **Runtime Configuration:** In the Colab notebook, go to "Runtime" -> "Change runtime type". Select "T4 GPU" and click "Save".

4. **Access Authorization:** Upon running the Generative Search module, you will be prompted to grant access to your Google Drive ("MyDrive"). Allow access to ensure seamless functionality.

5. **Utilizing the Generative Search Module:** Once the setup is complete, execute the generative_search(input()) function. Input a single question when prompted, and the Generative Search module will provide a contextually relevant response.

Note: Ensure that you have granted the necessary permissions and have the required files in the designated locations before executing the notebook. This setup guarantees a smooth and efficient experience with the Generative Search System. These instructions provide a comprehensive guide for users, outlining the steps from preparation to utilizing the Generative Search functionality.

Evaluation of Generative Search System

In this section, we provide a comprehensive evaluation of our Generative Search System, examining both quantitative and qualitative aspects to gauge its accuracy and relevance in responding to user queries related to the "CS 410 Text Information Systems" course contents.

1. **Quantitative Evaluation:** To assess the quantitative performance of the Generative Search System, a cosine similarity analysis was executed on the initial 25 questions related to the "CS 410 Text Information Systems" course. The cosine similarity scores between the "response" and "generated_response" were reported with three decimal points (e.g., 0.789). Responses with a score above the 0.5 threshold were considered accurate. For the second set of 25 unrelated questions, a score of 1 was assigned if the "generated_response" matched the predefined response, and 0 otherwise. The results, documented in a Quantitative_Evaluation.csv, included four columns: "question," "response,"

"generated_response," and "cosine similarity." Remarkably, the evaluation revealed 100% accuracy, with all comparison scores surpassing the 0.5 threshold.

2. **Qualitative Evaluation:** In the realm of qualitative assessment, a meticulous manual review of the 50 questions and responses was conducted by the team. The focus was on gauging the accuracy and relevance of the "generated_response" in comparison to the "response". "Qualitative_Evaluation.csv" file, generated within the "generative_search_development_evaluation.ipynb" notebook, encompassed four columns: "question," "response," "generated_response," and "is_generated_response_relevant?" The latter column contained binary values (0 or 1) indicating the relevance of the generated response, which was assigned by manual review of generated_response. Out of the 50 generated_response records, a staggering 48 were deemed relevant. It's noteworthy that the 2 instances of non-relevance were found in the subset of the first 25 questions related to the "CS 410 Text Information Systems" course.

While the system exhibited outstanding performance, achieving 100% accuracy in quantitative evaluation and 96% in qualitative evaluation, acknowledging the potential for model hallucination is crucial. Adjustments to the temperature parameter of the Dolly-v2-3b LLM during response generation could offer avenues for further enhancement. Experimentation with these parameters presents an opportunity to refine the system's output quality.

This evaluation not only validates the Generative Search System's quantitative accuracy and qualitative relevance but also highlights possibilities for refinement and optimization. While achieving impressive results, ongoing efforts to enhance the system's capabilities remain integral to its continuous improvement.

Contribution of each team member

In our collaborative project, each team member played a crucial role in ensuring its success:

1. **Timothy Cannella:**
    a. Led Track 1, overseeing Data Collection and Q&A Dataset Creation.

    b. Also, took charge of designing the Generative Search System in Track 2.

    c. Shared responsibilities with Luke Freitag for preparing the Presentation in Track 4.

2. **Mohammad Tamim:**
    a. Took the lead in Track 2, driving the Development of the Generative Search System, encompassing tasks such as development of build_vectors.ipynb, generative_search_development_evaluation.ipynb, and generative_search.ipynb.

    b. Contributed to Track 4, taking responsibility for preparing both the Progress Report and the Final Report.

3. **Luke Freitag:**
    a. Managed Track 2, Testing of generative Search system.

b. Took the lead in Track 3, driving the Results Evaluation, encompassing tasks such as Evaluation Sample Set Creation, Quantitative Evaluation, and Qualitative Evaluation.

c. Shared responsibilities with Timothy Cannella for preparing the Presentation in Track 4.

Each team member's dedicated more than 20-hour effort in their respective tracks significantly contributed to the overall success of our project. This collaborative approach ensured a well-rounded and effective generative search system.

Conclusion

In conclusion, our team's Generative Search project stands as a testament to the synergy of cutting-edge technologies in information retrieval, natural language processing, and machine learning. Through meticulous planning and collaborative efforts, we've successfully developed a sophisticated system that overcomes the limitations of traditional search engines. Leveraging RAG with LLMs, our Generative Search system offers users contextually accurate responses to queries related to the "CS 410 Text Information Systems" course. Our planning, development, and evaluation tracks ensured a methodical approach, resulting in a robust system that demonstrated 100% accuracy in quantitative evaluation and 96% accuracy in qualitative evaluation. Each team member's contributions were instrumental, covering various aspects from data collection to system design, development, testing, and evaluation.

Looking forward, there are exciting possibilities for enhancing our Generative Search system. Future work could focus on building a hybrid retrieval system that incorporates both semantic and syntactic search methods. This could further refine the system's ability to understand and respond to a broader range of user queries. Additionally, addressing the challenge of reducing LLM hallucination is an avenue worth exploring. Experimentation with parameters such as temperature during response generation with the Dolly-v2-3b LLM could lead to improvements in response quality. These advancements would contribute to the continual refinement and optimization of our Generative Search system, ensuring its adaptability and effectiveness in addressing user needs.