CSE 4/535-Introduction to Information Retrieval - Fall 2023

Project 3 Requirements

Due Date: 10th December 2023, 11:59 PM EST

Introduction

The final project aims to integrate the skills and concepts learned in the first two projects into a comprehensive Information Retrieval (IR) chatbot.

Project 1: Indexing and Crawling - Focused on gathering data from Wiki Pages, students learned to index this data effectively using Apache Solr.

Project 2: Scoring - Explored the mechanics of building your own inverted index from scratch and retrieving documents using AND queries.

Project 3: End-to-End IR Chatbot.

There are two different topics for the final project. Select any one of them (Topic 2 contains additional bonus points if attempted)
Difficulty Level: Topic 1 is easier compared to Topic 2

Topic 1: Wikipedia Question Answering System

This project aims to create a wiki Q/A Chatbot; the user can chit-chat with the bot or ask for information about a specific topic. The bot should be able to satisfy the user with relevant information. You need to implement various different components for this topic. The components are discussed below.

A. **Wiki Scraper**: You should implement a similar scrapper that you have already used in Project 1. The only difference this time is you need at least 50000 documents instead of 5000 total documents, and from each topic, you should have 5000 documents.
    a. Scrape data from Wiki related to the following Topics. Some subtopics are provided as suggestions.
        i. **Health**: Common diseases, global health statistics, mental health trends…
        ii. **Environment**: Global warming, endangered species, deforestation rates…
        iii. **Technology**: Emerging technologies, AI advancements…

        iv. **Economy**: Stock market performance, job markets, cryptocurrency trends…
        v. **Entertainment**: Music industry, popular cultural events, streaming platforms...
        vi. **Sports**: Major sporting events, sports analytics...
        vii. **Politics**: Elections, public policy analysis, international relations…
        viii. **Education**: Literacy rates, online education trends, student loan data…
        ix. **Travel**: Top tourist destinations, airline industry data, travel trends…
        x. **Food**: Crop yield statistics, global hunger, and food security…
    b. Get a minimum of 5000 documents for each of these topics listed above
    c. Make sure all the 5000 documents from each topic are unique

B. **Chit-chat**: For the chit-chat component, you can train a small chit-chat Language Model (LM), host a pre-trained LM as an API, or implement it in other ways. DO NOT MAKE IT RULE-BASED. Here are some references which will help you implement your own chit-chat model. The model should not stop chatting unless the user intends to.
    a. Chit-chat Dataset: https://github.com/BYU-PCCL/chitchat-dataset
    b. ChatterBot: https://chatterbot.readthedocs.io/en/stable/
    c. DialoGPT: https://huggingface.co/microsoft/DialoGPT-medium
    d. BlenderBot: https://huggingface.co/facebook/blenderbot-400M-distill

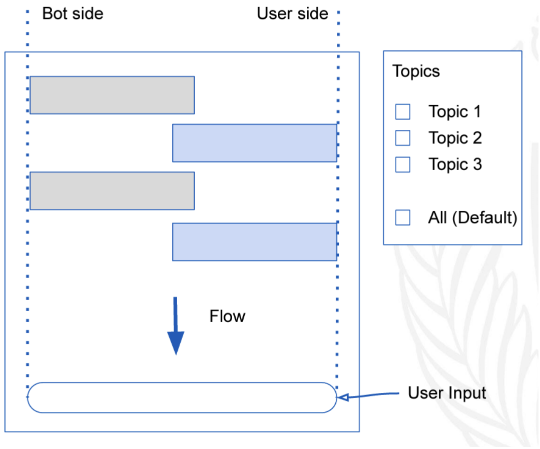
C. **Topic Analysis**: Given a user query you should be able to classify the topic (if not provided already) from the query and restrict your chat in that specific topic. You will get bonus points if you can implement multi topic chat.

D. **Wiki Q/A Bot**: Given the user query and the classified topic you should retrieve, top k documents and post process it to display a summary to the user. The post-processing implementation is upto you. You can use a generative model on top of the retrieved documents to generate a coherent summary (Bonus points will be there for this implementation)

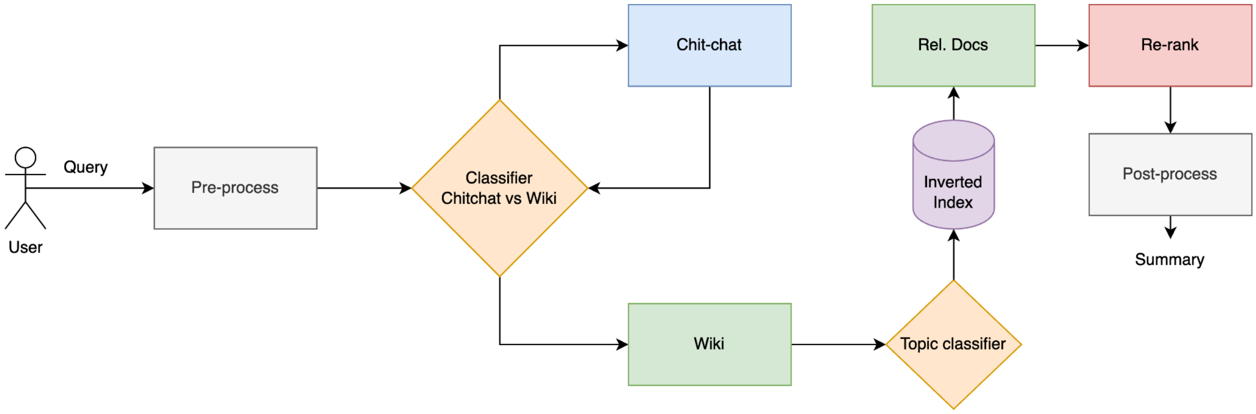
E. **Exception Handling**: Your chatbot might often fail to understand the user need, it should NOT crash. You should implement exception handling. Here are some references:
    a. https://www.chatbot.com/academy/chatbot-designer-free-course/error-messages/
    b. https://docs.python.org/3/tutorial/errors.html

F. **Visualization**: You should create a User Interface (UI) to converse with your chatbot. **You must host it as a Web App. It should NOT be in your Localhost.** There should be detailed analysis/visualization of chats, across topics, entities etc.

G. **Chat UI**



Overview of the Architechture



Topic 2: Retrieval Augmented Generation (RAG) on Novels (Guaranteed +5 Bonus)

This project aims to create a Q/A System on a collection of Novels; the user can chit-chat with the bot or ask for information about a specific Novel. The bot should be able to satisfy the user with relevant information. You need to implement various different components for this topic. The components are discussed below.

A. **Novels/ Story Book Data**: You should gather data from the Novels. You need at least 10 Novels. One of the novels that should definitely be there is:
   a. The Adventures of Sherlock Holmes by Arthur Conan Doyle
   b. You can find more here: https://www.gutenberg.org/
B. **Chit-chat**: For the chit-chat component, you can train a small chit-chat Language Model (LM), host a pre-trained LM as an API, or implement it in other ways. DO NOT MAKE IT RULE-BASED. Here are some references which will help you implement your own chit-chat model. The model should not stop chatting unless the user intends to.
   a. Chit-chat Dataset: https://github.com/BYU-PCCL/chitchat-dataset
   b. ChatterBot: https://chatterbot.readthedocs.io/en/stable/
   c. DialoGPT: https://huggingface.co/microsoft/DialoGPT-medium
   d. BlenderBot: https://huggingface.co/facebook/blenderbot-400M-distill
C. **Topic Analysis**: Given a user query you should be able to classify the Novel (if not provided already) from the query and restrict your chat in that specific novel. You will get bonus points if you can implement multi novel chat.
D. **Q/A Bot**: Given the user query and the classified novel you should retrieve, top k documents and post process it to display an answer to the user. The post-processing implementation is upto you. You **SHOULD** use a generative model on top of the retrieved documents to generate a coherent answer/ response.

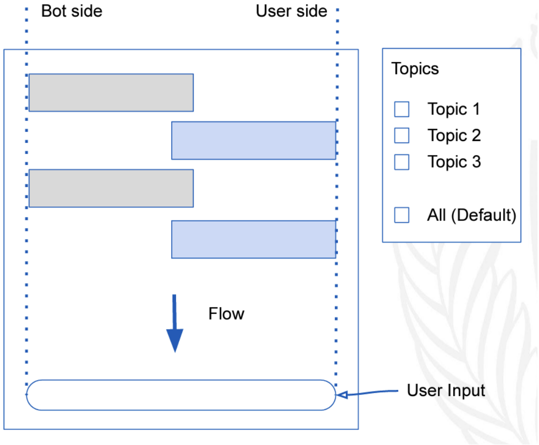
E. **Exception Handling**: Your chatbot might often fail to understand the user need, it should NOT crash. You should implement exception handling. Here are some references:

   a. https://www.chatbot.com/academy/chatbot-designer-free-course/error-messages/

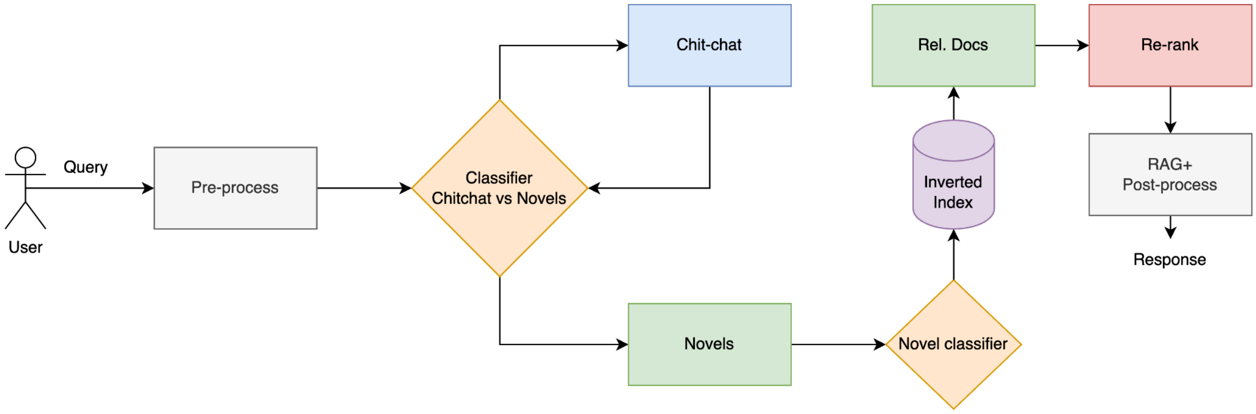
   b. https://docs.python.org/3/tutorial/errors.html

F. **Visualization**: You should create a User Interface (UI) to converse with your chatbot. **You must host it as a Web App. It should NOT be in your Localhost**. There should be detailed analysis/visualization of chats, across topics, entities etc.

G. **Chat UI**



Overview of the Architechture



Visualization Ideas for Both the Topics

1. Save N number of conversations for analytics and visualization.
2. Main purpose is to understand how well your chatbot is able to converse in the defined topics and how diverse are the responses.
3. Understand which topics the chatbot is able talk more about and where most of the errors are occurring. Also analyse the user queries and what type of queries are fetching more relevant utterances.

4. Based the previous point can we group certain queries? Do query reformulation increases the chat coherency?
5. Be creative and come up with your own ideas.

Group

1. You need to form your own groups of at max 3 members.
    a. Recommended: 3 or atleast 2 (1 UI, 2 backend)
    b. If you are going solo, you will be eligible for bonus if you can implement everything
2. Sign-up your team using the Google Form posted on Piazza before 20th Nov, 8 PM EST.
3. You are allowed to share your data within the group.
4. You are free to collect more data.

Final Deliverables

- A short demo video (at most 3 minutes)
- A working web application URL hosted on GCP
- A short report detailing all work done and member contributions.
    - You can use the double column ACL 2022 Latex template.
    - You can also use word, if you are not comfortable with Latex.
- The report should contain the following broad sections: (i) Introduction, (ii) Methodology (iii) Sample screenshots (iv) Work breakdown by teammates (v) Conclusion

Submission

You will get Bonus (+5) if you can submit a demo video of your working chatbot by 3rd December, 5 PM EST. The best groups will demo their work on 4th and 6th December. We will release a demo sign up sheet later.

You are required to submit the hosted URL in a text file (url.txt). You should upload everything in Brightspace. You need to create a zip file containing the following

1. The URL
2. The codebase in a directory named ‘src’, which contains all codes (both frontend and backend)
3. The Report (PDF)
4. Demo Video (mp4, mkv, etc)

Grading (30 points + 10 Bonus)

This project is worth **10 points**, and these are distributed as follows:

Criterion	**Points**
UI	6
Basic Chit Chat	6
Topic Classifier	3
Q/A Bot	7
Exception Handling	3
Visualization	5
Bonus	5+5

Academic Integrity

Academic integrity policies will be taken very seriously. If similarities are found more than a threshold value(which will not be disclosed) with other submissions, you will get a **ZERO** score. If you were found continuously arguing with us regarding your grades(unless there is a fault on our side), your grade would be reduced by up to 50% of your original score, and you would be reported to the department.

DEADLINE: This project has a **hard** deadline. You can still submit it with one day delay and lose 50% marks (You will be graded out of 15). If you submit after that you will get zero.