CSCI 4141 Information Retrieval
Assignment 4: Retrieval Augumentation Generation and Language Models

July 25, 2025

- Programing language: Python (Jupyter IPython Environment)
- **Due Date:** Refer to the course syllabus

Marking scheme and requirements: Full marks will be given for (1) working, readable, reasonably efficient, documented code that achieves the assignment goals, (2) for providing appropriate answers to the questions in the Jupyter notebook pushed to Birhgtspace.

Please cite all the resources used – the websites with source code you used should be listed in the submitted Jupyter notebook.

What/how to submit your work:

1. All your code should be included in a notebook.
2. For questions that require you to write textual answers use the Markdown option provided by Jupiter Notebook. Ensure that all answers are appropriately numbered as per their corresponding questions.
3. **Submit the completed Jupyter Notebook to Brightspace.**

Task Overview

This assignment focuses on exploring and implementing advanced concepts and techniques in information retrieval. The primary objectives are to develop a semantic search engine using Retrieval Augumentation Generation, and Language Models

Q1. Setting up the libraries and environment (1 Mark)

To get started with this assignment, you will first need to set up your development environment. This includes installing the required libraries in your Jupyter notebook. This task would help you with that. Installing dependencies is a continuous task that spans across the questions; however, in the end, you should have all the installation commands in the notebook itself and, hence, should have the answer to this.

(a) Install the required libraries in the notebook. Your assignment notebook should be able to install all the libraries using the commands in the notebook. (1 mark)

Q2. Data Preprocessing and Model Selection (7 Marks)

In this section, you will focus on preparing the dataset for the RAG model and selecting an appropriate pretrained language model. Data pre-processing involves loading the dataset, tokenizing the text, splitting it into chunks, and creating a vector store for efficient retrieval.

1. Use any dataset and perform necessary pre-processing steps. (2 marks)
2. Tokenize the text data using an appropriate tokenizer for the chosen pretrained language model. (2 marks)
3. Split the tokenized data into chunks suitable for indexing. (1 mark)
4. Create a vector store using a suitable library (e.g., FAISS, Pinecone) to efficiently store and retrieve the chunked data. (2 marks)

Q3. Implementing RAG using LangChain for different queries (8 Marks)

In this section, you will select a model selection requires considering factors such as pre-training methods and architecture to choose a language model that aligns well with the task at hand. You will then implement the RAG model using the LangChain framework and formulate meaningful queries, and demonstrate the effectiveness of the RAG model by presenting the generated responses.

1. Explain the RAG pipeline, describing all the components and their roles. (2 marks)
2. Choose a pretrained language model that aligns well with the task at hand. Provide a brief justification for your choice for the upcoming task (1 mark).
3. Set up the RAG pipeline using LangChain, integrating the chosen pretrained language model and the vector store. (2 mark)
4. Formulate meaningful queries related to the dataset and use the RAG pipeline to generate responses. (2 marks)
5. Demonstrate the effectiveness of the RAG model by presenting the generated responses and highlighting their relevance to the queries. Provide a brief analysis of the results. (1 mark)

Q4. Modify and evaluate the different components of RAG (7 marks)

In this section, you will experiment with different modifications to the RAG pipeline to improve its performance. This includes trying different retrieval techniques, fine-tuning the language model, modifying the prompt template, adjusting the number of retrieved documents, and evaluating the modified pipeline using examples. Finally, you will present a comparative analysis of the original and modified RAG pipelines.

1. Experiment with different retrieval techniques (refer to Langchain documentation for more details) within the RAG pipeline and compare their impact on the generated responses. (2.5 marks)

2. Modify the prompt template used in the RAG pipeline to provide more context or guidance for generating responses and analyze the resulting improvements. (1 mark)

3. Adjust the number of retrieved documents or passages used by the RAG model and examine how it influences the relevance and coherence of the generated responses. (1 mark)

4. Present a comparative analysis of the original RAG pipeline and the modified versions, highlighting the improvements in the generated responses using qualitative examples. (2.5 mark)

Q5. Selecting and implementing a pretrained model for a new task (8 marks)

In this section, you will choose a new task and implement it using a pretrained model that is different from the one used in the tutorial or previous questions. The pretrained model should have been trained using either Supervised Fine-Tuning (SFT), autoregressive training, or trained on Reinforcement Learning from Human Feedback (RLHF)(You can use models on Huggingface for completing this task).

1. Select a new task (e.g., text classification, named entity recognition, question answering) that is different from the tasks covered in the tutorial or previous questions. (3 mark)

2. Choose a pretrained model that is suitable for the selected task and is trained using either SFT, autoregressive language modeling, or RLHF. The model should be different from the ones used in the tutorial or previous questions. (2.5 mark)

3. Implement the selected task using the chosen pretrained model and relevant libraries or frameworks. (2.5 marks)

Validation

(a) Make sure your notebook runs without generating exceptions by restarting it and running all code cells. This can be done by Choosing Kernel −> Restart & Run all. You should see no exceptions.

(b) Make sure to add relevant comments to your code

(c) Make sure to add the answers to short answer-type questions in the notebook.