National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

CS416: Large Language Models

Class: BESE-12

Project Details

Date: 8th April 2025

Instructor: Prof. Dr. Faisal Shafait

Introduction

In this project, you will design a Large Language Model (LLM)-based solution to enhance customer service for a local bank. The goal is to convert a curated set of anonymized customer interaction documents into a responsive AI-driven assistant capable of accurately handling customer inquiries, generating coherent and context-aware responses, and maintaining a high standard of data privacy and trust.

Dataset Description

You will be provided with a dataset from a fictional bank. Your responsibility includes parsing and preprocessing the dataset, which may be in formats such as JSON, CSV, or plain text. Dataset is available on LMS.

Languages and Tools

You are free to use any languages and frameworks as you desire. You are encouraged to use this opportunity to learn new tools and technologies. This project provides you with a unique opportunity to learn the interactions between different tools.

Project Requirements

The requirements of the project are described below. Your system must contain the following features:

1. **Data Ingestion & Preprocessing:** Read and sanitize all documents from the local bank's dataset. Implement anonymization steps if they are not already performed. Handle tokenization, lowercasing, or other text-cleaning tasks to prepare data for LLM workflows.
2. **Large Language Model Selection:** You may use any open-source model such as Llama, T5, DeepSeek or a similar transformer-based architecture. Use of commercial models such ChatGPT is not allowed. Ensure the chosen LLM can handle your dataset effectively and can be integrated with your system for fine-tuning or prompt engineering. You will also justify why you choose the model.
   Note that the scope of this project is limited to models with 6 billion parameters. Please select models accordingly.
3. **Embedding & Indexing:** Create an embedding-based index. Store vector embeddings for each document (or chunk of text) so relevant content can be retrieved swiftly.
4. **Model Fine-Tuning & Inference**: Fine-tune your chosen LLM on the local bank data to improve domain-specific language understanding. For queries, retrieve relevant document segments using the embedding index, then have the model generate or synthesize an answer.
5. **Prompt Engineering**: Tailor your LLM to simulate helpful, caring interactions typical of customer service chats. The model should give domain specific answers and should gracefully handle out-of-domain questions.

6. **Real-Time Updates:** Allow for new documents (e.g., fresh FAQ entries or updated bank policies) to be seamlessly added and indexed. Any new information should instantly become available to customers seeking the latest updates. Let's say a user adds a new document into the system (following the dataset format), he or she shall be able to ask any question and LLM should be able to generate the response based on new added document. You can also provide a User Interface to add new data/articles/documents/items etc.
7. **Performance and Reliability:** Aim for minimal lag when dealing with multi-word or complex queries which is important for delivering a smooth customer experience. Demonstrate that your solution can scale effectively to large datasets without compromising service quality.
8. **Using GIT for team collaboration**: You should not complete the project and then upload it once on bitbucket or github. Instead, you should keep pushing the updates as commits as you progress through the project. All group members should make commits of their contributions. You can register to make a private repository on Github.
9. **System Interface**: Develop a clear and welcoming user interface where customers can submit questions, view responses, and upload additional information (documents). Keep the design simple but reassuring, so users know they're interacting with a reliable support tool.
10. **Application of Guard Rails**: Implement content filtering and policy enforcement so the LLM does not share sensitive or disallowed information. Include a mechanism to manage potentially harmful or off-topic requests (e.g., refusing to provide disallowed details or redirecting the user). Proactively detect and mitigate "jailbreaking" or "prompt injection" attempts, in which users try to manipulate the model to bypass restrictions or reveal confidential data.

Grading Criteria:

The maximum score you can get is 20. Grading criteria is described in the table below. Marks will be given to each group member based on their contribution and performance in the viva.

Criteria	**Total Points**
Data Preprocessing: Thoroughly removing or masking PII	2
Vector Embeddings: Fast and accurate retrieval of relevant info	3
Prompt Engineering (model handles domain specific and out-of-domain queries accurately and gracefully)	2
Implementation of Guard Rails (model proactively detect and mitigate jailbreaking, prompt injection attempts, does not share sensitive or disallowed information and is not hallucinating)	4
Model responses are user friendly with minimal lag	2
Ability to add documents directly from the user interface	2
Code is of high quality (follows the conventions and code style guides) and well commented	2
System has an intuitive user interface.	1
GIT is used for collaboration.	2

Timeline:

The project has three deliverables, as mentioned below:

1- Project Proposal

A short document that includes:

- List of group members.
- Languages and frameworks to be used.

Deadline: 14th April 2025

2-LLM Implementation

Here you must submit your code for implementing LLM. Provide a prototype system that returns initial, LLM-driven answers. You will also submit Architecture Diagram.

Deadline: 1st May 2025
Note: This submission will be evaluated as a Lab Project

3- Complete Submission

Final code covering ingestion, embeddings, model integration, guard rails, and user interface. Documentation detailing architecture, usage instructions, references

Deadline: 19th May 2025

4- Presentation, Demo and Viva

Deadline: 21st May 2025
Note: Presentation and Demo will be evaluated as Course Project and Viva will be evaluated as Lab Final

Project Discussion:

If you have any confusions regarding the project specification or want an idea of different tools and frameworks that can be used for your implementation, you can send an email at **"aymen.tasneem@seecs.edu.pk"**. Note that you must understand the algorithms and debug your code yourself.

Plagiarism Policy:

Plagiarism in any form will not be tolerated. You must complete the project on your own and provide credit where the credit is due. You will be graded based on what you implemented by yourself. If any form of plagiarism is detected, you will be assigned a grade of zero in your project.

Range of Complex Problem Solving:

	Attribute	Complex Problems
1	Preamble	This project cannot be resolved without an in-depth engineering knowledge.
2	Depth of analysis required	The student will conduct a Data Quality Assessment to evaluate whether the documents are complete, clean, and contextually relevant. Additionally, the student will analyze and determine an appropriate chunking strategy, deciding whether to split the text semantically or based on token limits.
3	Depth of knowledge required	Extensive knowledge will be needed regarding the inner working of large language models and the trade-offs associated therein. It would be necessary to refer to research papers to figure out the optimal approach to solving these problems.
4	Extent of applicable codes	Standard coding style guides are applicable depending on the languages/frameworks that are used for development.
5	Interdependence	The project will require students to develop many different components that rely on the output of other components. For example: This project consists of several interlinked components (e.g., data ingestion, embedding generation, query handling). Each module relies on the outputs of previous steps, and only by integrating these parts effectively can student deliver a cohesive and functional solution.

National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Range of Complex Engineering Activities:

	Attribute	Complex Activities
1	Preamble	The project will require the students to complete the following complex activities:
2	Range of resources	This project will require the use of several libraries for different tasks like model implementation , prompt engineering, guard rails and User Interface etc. Similarly it will require expert human resources in the aforementioned tasks.
3	Level of interaction	This project will have several components which will require interaction with each other. High level of interaction brings with it a lot of issues which will have to be solved. Majority of the new components will rely heavily on the output produced from the previous component to further process the data.
4	Innovation	An approach has to be developed for searching such that the performance does not deteriorate significantly on increasing the dataset size or the query length. This may require coming up with techniques and learning tools not previously encountered by the students.
5	Familiarity	Students must leverage their existing knowledge of data structures and algorithms—originally applied to tasks like basic document searches—and adapt it for LLM-based retrieval and ranking. Rather than simply returning relevant items, the focus now is on generating high-quality, context-aware responses, requiring students to extend their familiarity into new areas such as embedding, semantic ranking, and prompt engineering.