CS5342 Network Security

<u>Group-based Project</u>

1. Project Objective:

Create your own personalized bot utilizing the capabilities of a Large Language Model (LLM) to engage with your network security documents.

2. Your Task

Task1: OnDemand Professor Q&A Bot

- Your task is to build a Q&A Bot over private data that answers questions about the network security course using the open-source alternatives to ChatGPT that can be run on your local machine. Data privacy can be compromised when sending data over the internet, so it is mandatory to keep it on your local system.
- Your Q&A Bot should be able to understand user questions and provide appropriate answers from the local database, then the citations should be added **(must be accomplished)** if the response is from the internet, then the web references should be added.
- Train your bot using network security lecture slides, network security textbook, and the Internet.
- By using Wireshark capture data for Step 4's of the LLM workflow shown in Figure 1. Provide detailed explanations of the trace data. Also, Maintain a record of Step 1's prompt and its mapping to the trace data in Step 4's.

or

Task2: Quiz Bot

- Your task is to build a quiz bot based on a network security course using the open-source alternatives to ChatGPT that can be run on your local machine. Data privacy can be compromised when sending data over the internet, so it is mandatory to keep it on your local system.

- Two types of questions should be offered by the bot: randomly generated questions and specific topic questions and the answers should be pulled from the network security database. Train your bot using network security quizzes, lecture slides, network security textbook, and the Internet.
- The quiz must include multiple-choice questions, true/false questions, and open-ended questions.
- Finally, the bot should be able to provide feedback on the user's answers.

3. Deliverables and Submission:

- **09/08/2023,11:59 PM (Round 0)**

  1. Form a group **(no submission)**

     The project will be assigned as a group project. Six students will be considered a group for a project. Here is the link to enter your project group member names.

     FALL 2023 CS5342 PROJECT GROUP NAMES.xlsx

     It is recommended that one person in the group fills in the form to avoid multiple entries and submits project files on Blackboard. If you cannot get a group, contact the TA.

     The group members will collaborate to develop a project. Each member will be required to contribute and adhere to group objectives. The final product should be presented at the end of the semester.

  The professor will assign the task once the groups are formed.

- **10/13/2023, 11:59 PM (Round 1)**.

  1) Report (groupno__Task1report.docx or groupno__Task2report.docx)
     - In group reports include the proposal page along with the system framework (3-4 paragraphs of 10 lines)
     - The work of each team member should be detailed so that the unique contributions of each team member can be highlighted and substantiated.
     - Group Report Format
       a) The group report should be no more than **three 8.5x11 pages**, have a black font of 10-point type, have 1-inch margins, and be single-spaced.

b) Contribution Table of group members
    - Group member names should appear in the left column, one per row.
    - A corresponding bulleted list of significant group member accomplishments made or to be made should appear in the right column.

2) GitHub repository link
    - Set up a repository on GitHub for your project.
    - It is required that each member of the team be connected to the project GitHub repository as a contributor.
    - Submit the link to your GitHub repository via the BB.

- **11/10/2023, 11:59 PM (Round 2)**
    1. Prototype submission (groupno_prototype.zip)
        - Software setup, source code, database, adopted libraries, etc.
        - **For captured data: (Task1 only)**
            - Each group member captures 5 prompts and explains in detail the captured data with the help of a screenshot. Include a mapping between Step 1's prompt and Step 4's trace data.
            - submit student_name.docx and student_name.pcp

        - **README file:** In this file write the required environment, adopted libraries, the flow of execution, commands to run the code, the issue you are experiencing, and solutions, Suggestions, and Feedback
        - Include all the above in the group_no.zip and submit it via the BB.
    2. Prepare slides (groupno_round2.pptx)
        - The format of the slides is listed below.
    3. GitHub repository link
        - Update the repository on GitHub for your project.
        - Create a README file in the repository that includes clear instructions for submitting the projects. (Includes Project Description, documentation, System architecture, video, Prerequisite, requirements, Step by step instructions for

executions, features, describe training data and data formats etc.) refer references 10-12

- Submit the link to your GitHub repository via the Blackboard.

- **A tentative schedule: 11/13/2023 – 12/04/2023, (Round 3) (100 Points)**
  1. Demo and Q&A session for each group.
     - 10 -15 min for each group.
     - Schedule for the Demo and Q&A session will be displayed on 11/10/2023.
     - No submission, In-person meeting.

Presentation Format:

- Problem Title, group name, group member first and last names (up to 1 slide)
- Project statement (up to 2 slides)
- Proposed Solution (up to 2 slides)
- System requirements for the software prototype implementing the proposed problem solution (up to 2 slides)
  - Discuss the operating system, programming language, hardware, database, packages needed, etc.
  - Data collection
- Show the software architecture design diagram. (up to 2 slides)
- Discuss Pros and cons. (up to 1 slide)
- Discuss issues you are experiencing and solutions (up to 1 slide)
- Conclusion (up to 1 slide)
- Future direction (up to 1 slide)
- References (up to 1 slide)

4. Grading:

If you miss any of the requirements listed in this document, you may lose significant credits. e.g., you may lose significant points for the wrong file name format, for not submitting your team in a timely manner, or for missing components in a report and GitHub repository.

No late submission will be accepted. Read and follow this document carefully.

- **10/13/2023, 11:59 PM (Round 1). (100 Points)**
  1. Report (groupno_Task1report.mp3) - 50points
  2. GitHub repository link - 50 points

- **11/10/2023, 11:59 PM (Round 2).(200 Points)**
  1. Prototype submission (groupno_prototype.zip) - 50 points
  2. Presentation (groupno_round3.pptx) – 50 points
  3. GitHub repository link - 100points

- **A tentative schedule: 11/13/2023 – 12/04/2023, (Round 3) (100 Points)**
  1. Demo - 50 points
  2. Q&A - 50 points

  Evaluation criteria will be displayed after round 1 submission.

5. More about the LLM(Chatbot)

5.1 What is a large language model (LLM)?

A Large Language Model (LLM) is a type of artificial intelligence model designed to understand and generate human-like text. It is trained on vast amounts of textual data to learn the patterns, structures, and nuances of human language. LLMs are a subset of a broader category of models called natural language processing (NLP) models.

These models are typically based on deep learning techniques, specifically transformer architectures, which allow them to process and generate text with remarkable fluency and coherence. LLMs have the ability to perform a variety of language-related tasks, including:

1. Text Generation: They can produce coherent and contextually relevant text based on a given prompt or input.

2. Translation: LLMs can translate text from one language to another with a reasonable degree of accuracy.

3. Summarization: They can automatically generate concise and coherent summaries of longer pieces of text.

4. Question Answering: LLMs can answer questions posed in natural language by extracting relevant information from a given text.
5. Sentiment Analysis: They can determine the emotional tone or sentiment expressed in a piece of text.
6. Language Understanding: LLMs can analyze and understand the meaning of sentences and documents, including recognizing entities (names, places, etc.) and relationships between them.
7. Chatbots and Conversational Agents: LLMs can be used to create chatbots and virtual assistants that engage in human-like conversations.
8. Content Creation: LLMs are capable of generating articles, stories, poetry, and other forms of creative writing.

One of the most well-known examples of a Large Language Model is GPT-3.5 (Generative Pre-trained Transformer 3.5), developed by OpenAI. GPT-3.5 has 175 billion parameters, making it one of the largest and most powerful language models to date. It has garnered significant attention for its ability to produce coherent and contextually appropriate text across a wide range of tasks.

As much as ChatGPT is convenient, it has its tradeoffs. The fact that it requires you to send your data over the internet can be a concern when it comes to privacy, especially if you're using confidential documents. Additionally, it requires a constant internet connection, which can be an issue in areas with poor connectivity. Fortunately, there is an alternative. You can run your own local large language model (LLM), which puts you in control of your data and privacy.

Here, you will find out how to create a private Chatbot that interacts with your local documents, giving you a powerful tool for answering questions and generating text without having to rely on OpenAI's servers.

5.2 How to create your own LLM.

Build your own ChatGPT that interacts with your local documents, allowing you to generate text and answer questions without dependency on OpenAI. Everything will be stored and run on your own computer.

The following are the components you need to create a local language model that interacts with your documents.

1. **Open-source LLM**:
   These are small open-source alternatives to ChatGPT that can be run on your local machine. Some popular examples include Dolly, Vicuna, GPT4All, and llama.cpp. These models are trained on large amounts of text and can generate high-quality responses to user prompts.
2. **Embedding model**:
   An embedding model is used to transform text data into a numerical format that can be easily compared to other text data. This is typically done using a technique called word or sentence embeddings, which represent text as dense vectors in a high-dimensional space. These embeddings can be used to find documents that are related to the user's prompt. The Sentence Transformers library contains a rich variety of pre-trained embedding models.
3. **Vector database**:
   A vector database is designed to store and retrieve embeddings. It can store the content of your documents in a format that can be easily compared to the user's prompt. Faiss is a library that you can use to add vector similarity comparisons on top of other data stores. But there are also a few open-source vector databases that you can install on your computer including Qdrant, Weaviate, and Milvus.
4. **Knowledge documents**:
   A collection of documents that contain the knowledge your LLM will use to answer your questions. These documents depend on your application. For example, it can be a collection of PDF or text documents that contain your personal blog posts.
5. **User interface**:
   The user interface layer will take user prompts and display the model's output. This can be a simple command-line interface (CLI) or a more sophisticated web application such as Streamlit. The user interface will send the user's prompt to the application and return the model's response to the user.

4.3 Own LLM (Bot) workflow

The following preparations must be made before you can use your ChatGPT:

1. Create a list of documents that you want to use as your knowledge base

2. Break large documents into smaller chunks (around 500 words)
3. Create an embedding for each document chunk
4. Create a vector database that stores all the embeddings of the documents

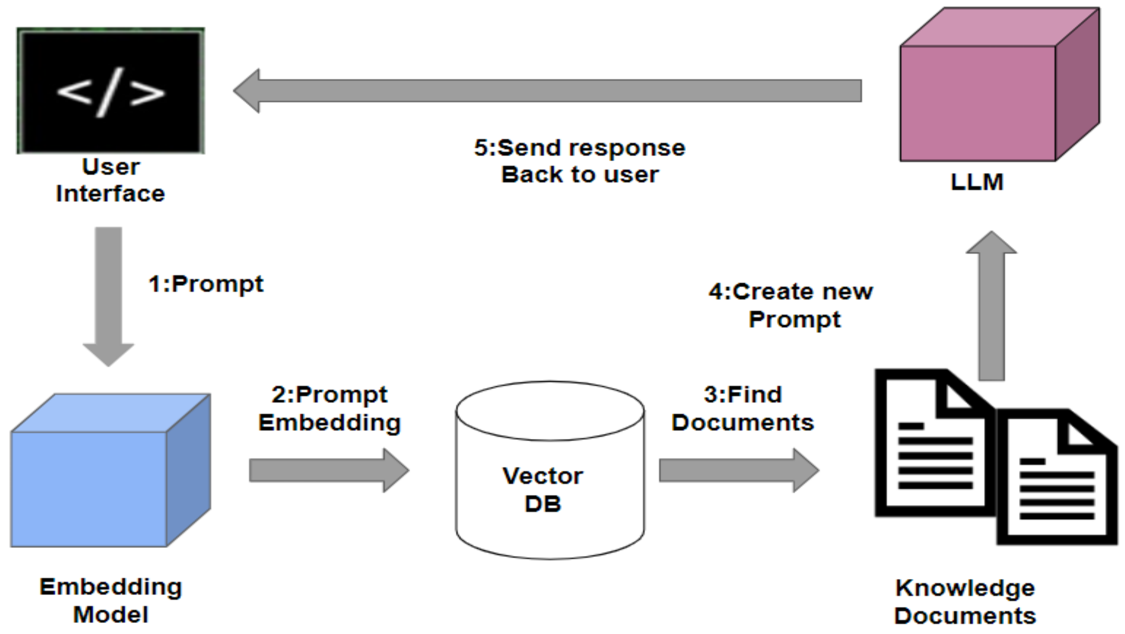

Figure 1: LLM workflow

Now that our knowledge base and vector database are ready, the Own LLM (Bot) workflow will be as follows:

1. The user enters a prompt in the user interface.
2. The application uses the embedding model to create an embedding from the user's prompt and send it to the vector database.
3. The vector database returns a list of documents that are relevant to the prompt based on the similarity of their embeddings to the user's prompt.
4. The application creates a new prompt with the user's initial prompt and the retrieved documents as context and sends it to the local LLM.
5. The LLM produces the result along with citations from the context documents. The result is displayed in the user interface along with the sources.

References:

1. How to Build Your Own AI Chatbot With ChatGPT API (2023) | Beebom
2. How to Build Your Own AI Chatbot With ChatGPT API: a Step-by-Step Ultimate Guide (softermii.com)
3. Build a Chatbot Based on Your Own Documents with ChatGPT | Step-by-Step Guide – Vasos Koupparis (vasos-koupparis.com)
4. Create a Powerful Chatbot with ChatGPT Using Your Documents (analyticsvidhya.com)
5. How to Train an AI Chatbot With Custom Knowledge Base Using ChatGPT API | Beebom
6. Create an Azure OpenAI, LangChain, ChromaDB, and Chainlit Chat App in Container Apps using Terraform - Microsoft Community Hub
7. How to create a private ChatGPT that interacts with your local documents - TechTalks (bdtechtalks.com)
8. What Product People Need To Know About LangChain | CommandBar Blog
9. Host a Llama 2 API on GPU for Free | by Yuhong Sun | Aug, 2023 | Medium
10. GitHub - Chainlit/chainlit: Build Python LLM apps in minutes ⚡
11. GitHub - Prographers/Slack-GPT: A Slack Bot implementation for integration with OpenAI ChatGPT. Supports GPT4 by default!
12. GitHub - VolkanSah/GPT-API-Integration-in-HTML-CSS-with-JS-PHP: A basic GPT conversation script designed to help you learn to interact with OpenAI's GPT technology. Includes best practices and a free security whitepaper.
13. How to Build a Personalized Unlimited Quiz App in Minutes: ChatGPT API Edition - DEV Community
14. How to Build a Multiple Choice Quiz with Chat GPT | by S713FF3N | Dev Genius