Assignment 2: Network Outage Chatbot and Prediction

I have upgraded our telecom network operations support system by replacing the basic rule-based chatbot with an AI-powered solution that leverages Retrieval-Augmented Generation (RAG) combined with Large Language Models (LLMs). In addition, I have implemented a predictive machine learning model that forecasts network outages based on historical outage data and relevant operational features. This document details my approach and solution for both tasks.

Objective #1: Transition to an AI-Powered Chatbot Using RAG and LLMs

System Design

Architecture Overview:

I designed the new chatbot system to combine several advanced components:

- **Text Extraction & Indexing:** I extract data from telecom reports (including tables and text) using a custom script. The text is split into overlapping chunks to capture both plain text and table data.
- **Embedding & Retrieval:** I use a pre-trained SentenceTransformer model (all-MiniLM-L6-v2) to compute embeddings for these chunks, which are then stored in a FAISS index for fast similarity search.
- **LLM for Generation:** For answer generation, I leverage Ollama's Llama 3.2 model via the CLI. This integration allows the chatbot to generate detailed, context-aware responses.
- **Reranking (Optional):** Although not implemented in this version, I considered using a reranker to optimize retrieval relevance.
- **Interface Adapter:** The system also includes logic to detect the type of interface (text or voice) and adjust the response style accordingly.

Integration:

The components interact as follows:

1. A user query is submitted via a Streamlit-based chat UI.
2. The query is embedded and used to retrieve the most relevant chunks from the FAISS index.
3. A dynamic prompt is constructed based on the extracted year (or query context) and the type of question (data extraction vs. anomaly/trend analysis).
4. This prompt is then passed to the Llama 3.2 model via the Ollama CLI to generate a detailed answer.
5. The final response is presented to the user through the chat interface.

Data Preparation

For the Chatbot:

- I used the TV Industry Report (which includes plain text, tables, and images) as the primary data source.
- Special care was taken to ensure that both the mixed-format content of the first 23 pages and the subsequent year-wise breakdowns (from page 24 onward) were included.

- I created a script (model.py) that extracts text using PyPDF2, splits it into overlapping chunks, computes embeddings, and builds a FAISS index.

For the Outage Prediction Model:

- I utilized a provided CSV file (network_output.csv) that contains historical network outage records.
- I preprocessed the data by parsing timestamps, calculating outage durations, converting textual equipment age into numerical values, and encoding categorical variables (maintenance history, weather, traffic load, etc.).

Model Selection

LLM Choice for Chatbot:

I selected Ollama's Llama 3.2 model for text generation because it offers strong reasoning capabilities, is open-source and locally deployable, and meets our cost and performance requirements in the telecom operations space. Its ability to generate detailed, context-aware responses makes it ideal for our needs.

Predictive ML Model:
For predicting network outages, I chose a RandomForestRegressor. This model is robust, capable of handling non-linear relationships, and provides useful feature importance rankings to highlight the key drivers of outages (such as equipment age, maintenance history, weather conditions, and traffic load).

RAG Implementation

I implemented the RAG system as follows:

- **Indexing:** I ran model.py to process the PDF and generate both the FAISS index and text chunks.
- **Retrieval:** At query time, the embedding model encodes the query, and I use FAISS to retrieve the most relevant text chunks.
- **Prompt Engineering:** I dynamically extract the year from the user's query. If the query asks for trends or anomalies, I instruct the model to compare the specified year with surrounding years. Otherwise, I instruct it to extract precise data for that year.

- **Generation:** The dynamic prompt is then sent to Ollama's Llama 3.2 (via a subprocess call using ollama run llama3.2), which generates the final answer.