NLP Maintenance Chatbot

Jace Bahls, Joseph Hsin, Krish Shah, Matthew Kurniawan,
Mehak Virdy, Parth Hathalia, Rushil Ramchand, Srushti Vaidyanathan

Acknowledgements: Thank you to Kali, Lauren, our V2X Mentor Doug, and the rest of the Data Mine staff for all their help this semester!

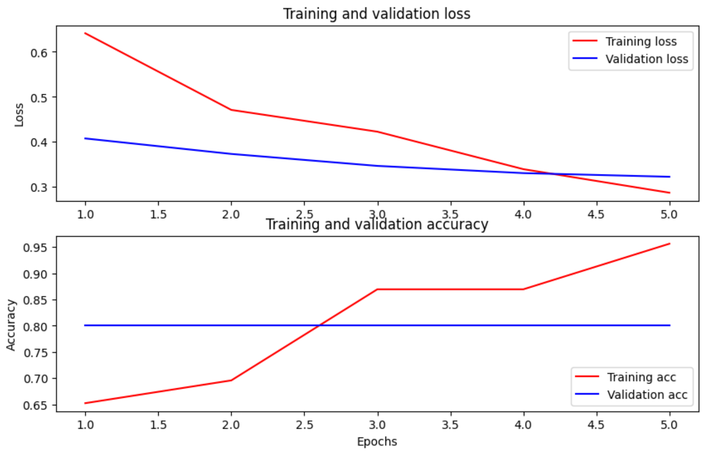
Introduction

- **Goal:** Develop a Chabot to increase accessibility of airport documentation for a troubleshooter application
- **Dataset:** Pan American Airways' (PanAm) manuals of procedures and maintenance

Research Methodology

- Web-scraped the documents using Python by fetching the photo scans' auto-transcripts with requests
- Cleaned and analyzed the content using NLTK (NLP library)
- Compared **two approaches** for document retrieval:
  1. Created a sample question index to train categorical classification with BERT
  2. Vectorizing the documents to create a quick-access database
- Decided on the latter approach to implement with retrieval augmented generation (RAG) with an LLM and a **Web App**
- **Control Flow:** Agile, Git

First Approach: BERT Model



- Set to classify a user question into two of PanAm's manual categories
- Used Python's TensorFlow library
- Inconsistent data quantities between categories led to irregularities in validation (overfitting and underfitting)

Fig 1. Model accuracy and loss overtime between "Airplanes" and "Business Records"

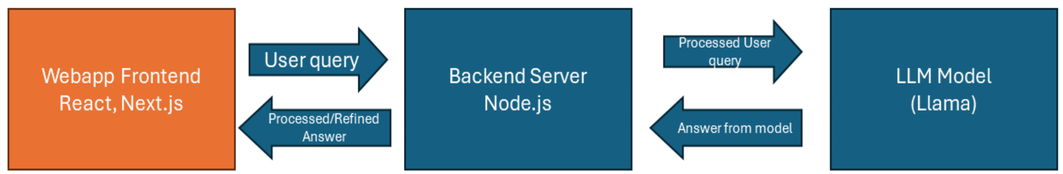


Fig 2. Framework outline between the Frontend and Backend

Second Approach: Vectorization

- Store the documents as embeddings with the Langchain Python Library
- Documents are considered vectors, and similarity can be drawn between them and a vectorized user query for retrieval

RAG

- Retrieval model uses the vectorized documents along with the user query to find an associated document or chunk of related information
- This context can be fed to an LLM for it to answer the original question
- Used an LLM on HuggingFace called Llama (hosted locally)

Web App

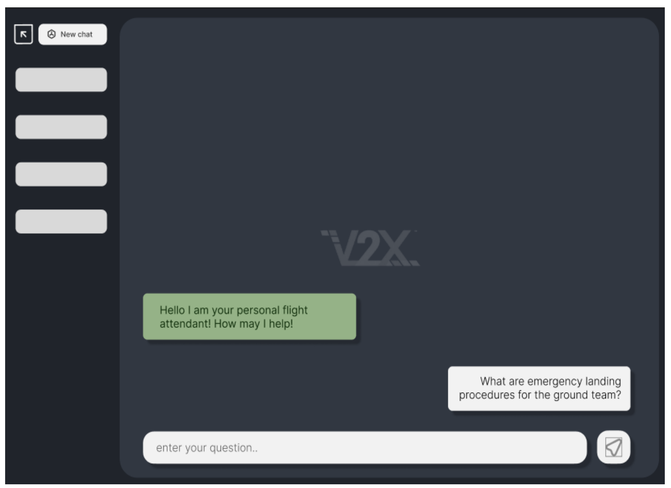


Fig 3. Web App User View – Interacting with the ChatBot

- Frontend is built with React
- Backend is built with the Express framework for Node.js
- A web server listens for a user query via websockets
- Receiving the query, the backend prompts document retrieval and RAG
- Response from LLM is relayed to the user

Accomplishments

- ✓ Utilized NLP techniques (Tokenization, Stemming, Lemmatization, Chunking) to clean and familiarize ourselves with the dataset
- ✓ Wrote Python Scripts for web-scraping, data processing, and machine learning.

Future Goals

- Re-visit the BERT model implementation with an emphasis on data cleanliness/management
- Customize model response based on the role of the user (for added retrieval and LLM context)
- Improve readability/reliability of Chatbot responses

Conclusions

- Reoccurring problem was data limitations related to quality (e.g. cleanliness of text) and quantity (e.g. more user questions needed for effective binary classification)
- We had to overcome these issues by implementing additional NLP techniques and reevaluating our solutions
- We learned how to use various Python libraries to interpret and use a public dataset for direct application in ML
- Using RAG and an LLM, we created a functional question-answering Chatbot