Voice-Activated Teaching Assistant in Persian and Hindi

Project Advisor: Dr Vahid Behzadan

Team Members:

- Bahareh Arghavani Nobar
- Devnath Reddy Motati
- Hawar Dzaee

WEEK	TASK	Status
WEEK1(JAN22-FEB5)	Research and choose a topic for Capstone Project	DONE
WEEK2(FEB6-FEB9)	Data Collection	DONE
WEEK3(FEB9-FEB12)	Data Preparation	DONE
WEEK4(FEB13-FEB19)	-Fine Tune XTTS for Persian, -Finding Kurdish S2T model, -Addressing the shortcoming of tts models in kurdish language	IN PROGRESS
WEEK5(FEB20-FEB26)	RAG System Implementation	IN PROGRESS
WEEK6(FEB27-MAR11)	Integrate all the system components	
WEEK7(MAR12-MAR31)	Test integrated voice assistant	
WEEK8(APR1-APR15)	Deployment	
WEEK9(APR16-APR30)	Performance Evaluation	

Report on Fine-Tuning XTTS v2 for Persian Audiobooks

Overview

Bahareh embarked on a project to fine-tune the XTTS v2 model, an open-source text-to-speech system, specifically for Persian audiobooks. During the course of the project, several challenges and insights emerged, culminating in a pivot towards a more suitable dataset and methodology.

Initial Challenges

The project encountered a significant hurdle when Bahareh attempted to fine-tune the XTTS v2 model. An error was reported: "AssertionError: :exclamation: len(DataLoader) returns 0. Make sure your dataset is not empty or len(dataset) > 0." This issue suggested that the dataset was not properly configured or recognized by the DataLoader, a critical component in training neural networks.

To address this, Bahareh considered modifying the `tokenizer.py` to accommodate 250 additional tokens. However, a Medium post on XTTS v2 advised against such alterations, emphasizing the importance of adhering to recommended dataset structures and parameters.

Report on Fine-Tuning XTTS v2 for Persian Audiobooks

Medium

Research and Solution

Further research into the XTTS v2 model led Bahareh to a Medium article which highlighted specific requirements for the dataset used in fine-tuning XTTS. The article recommended using audio chunks ranging from 3-6 seconds with a 24KHz sample rate. For compatibility with Google's Speech API, it was also necessary to ensure the audio was mono-channel.

Open in Colab

Acting on this advice, Bahareh recreated the audiobook dataset to meet these specifications. The process involved converting the audiobook audio to WAV format at a 24KHz sample rate, chunking it into 5-second segments, and evaluating transcription quality using Whisper and Google's Speech API. The decision was made to proceed with Google's Speech API based on the superior quality of its transcriptions.

Training and Challenges with the Original Dataset

The training results, available at Bahareh's Wandb.ai project page, revealed issues with the audio quality. It was determined that the fantasy-themed audiobook, characterized by expressive and emotional narration, was not compatible with the XTTS architecture. The model, designed to extract speaker features, produced distorted audio when processing the emotionally rich content.

Report on Fine-Tuning XTTS v2 for Persian Audiobooks

Strategy Shift

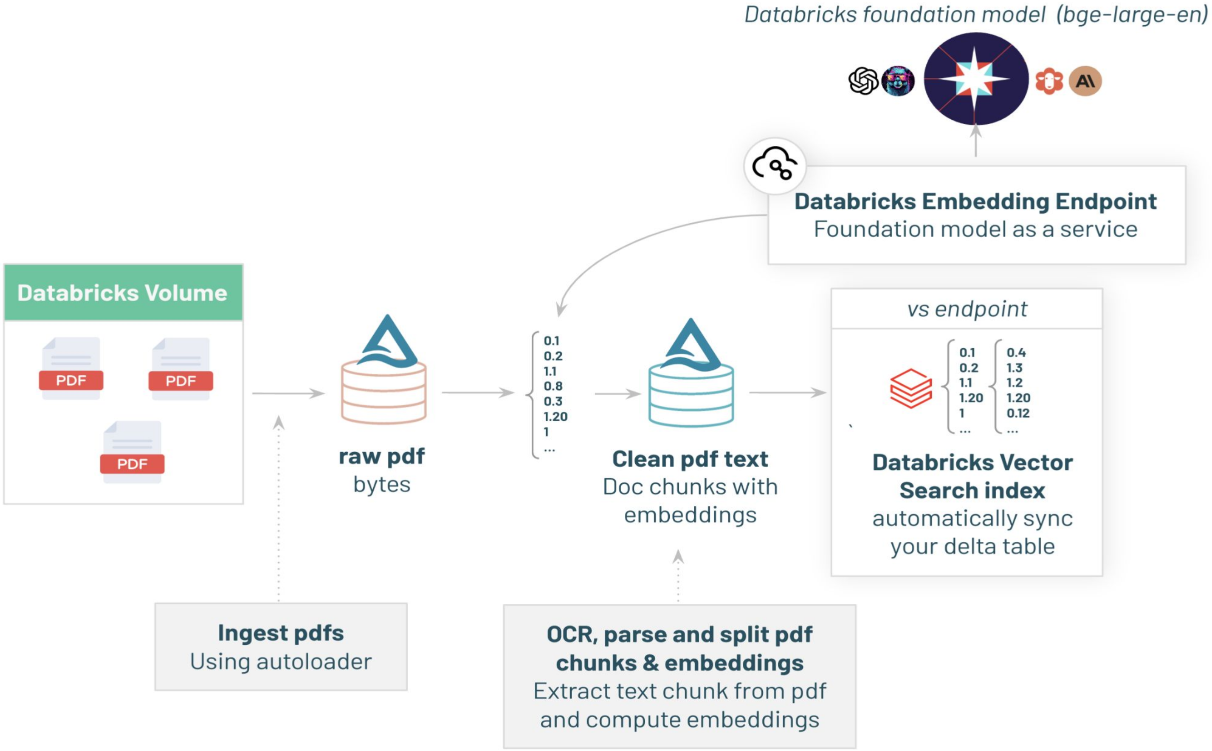
Recognizing the limitations of the initial approach, Bahareh decided to pivot towards a dataset featuring mono-speaker newsreader audio, characterized by less emotion. This decision was informed by the XTTS model's architecture, which is more suited to neutral, consistent speech patterns. Additionally, Bahareh expanded the model's `vocab.json` to include Persian tokenization, drawing upon resources from the Hugging Face model (bolbolzaban/gpt2-persian).

New Training Approach

The new strategy involved training the XTTS v2 model from scratch using the LJSpeech dataset, followed by fine-tuning on a Persian News dataset, and eventually expanding to Kurdish and Hindi datasets. This approach aims to leverage the XTTS model's capabilities while accommodating the linguistic and acoustic characteristics of the target languages.

RAG IMPLEMENTATION ON AZURE DATABRICKS

- Use autoloader to load the binary PDFs into our first table.
- Use the `unstructured` library to parse the text content of the PDFs.
- Use `llama_index` or `Langchain` to split the texts into chuncks.
- Compute embeddings for the chunks.
- Save our text chunks + embeddings in a Delta Lake table, ready for Vector Search indexing.

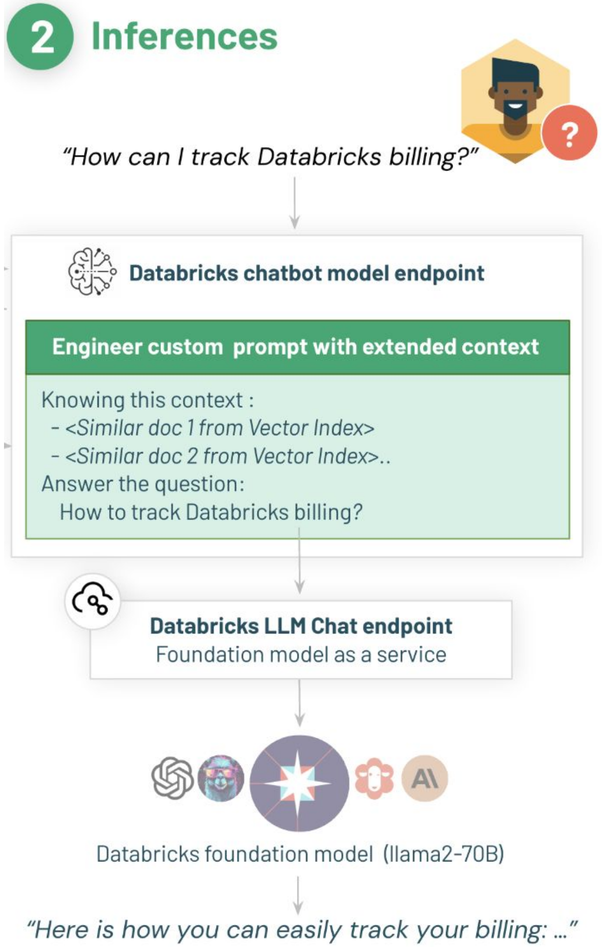


Databricks foundation model (bge-large-en)
Databricks Embedding Endpoint
Foundation model as a service
Databricks Volume
PDF
PDF
PDF
raw pdf
bytes
0.1
0.2
1.1
0.8
0.3
1.20
1
...
Clean pdf text
Doc chunks with embeddings
vs endpoint
0.1
0.2
1.1
1.20
1
...
0.4
1.3
1.2
1.20
0.12
...
Databricks Vector Search index
automatically sync your delta table
Ingest pdfs
Using autoloader
OCR, parse and split pdf chunks & embeddings
Extract text chunk from pdf and compute embeddings

- Build a complete chain supporting a chat history, using llama 2 input style

- Add a filter to only answer Databricks-related questions

- Compute the embeddings with Databricks BGE models within our chain to query the self-managed Vector Search Index



2 Inferences

"How can I track Databricks billing?"

Databricks chatbot model endpoint

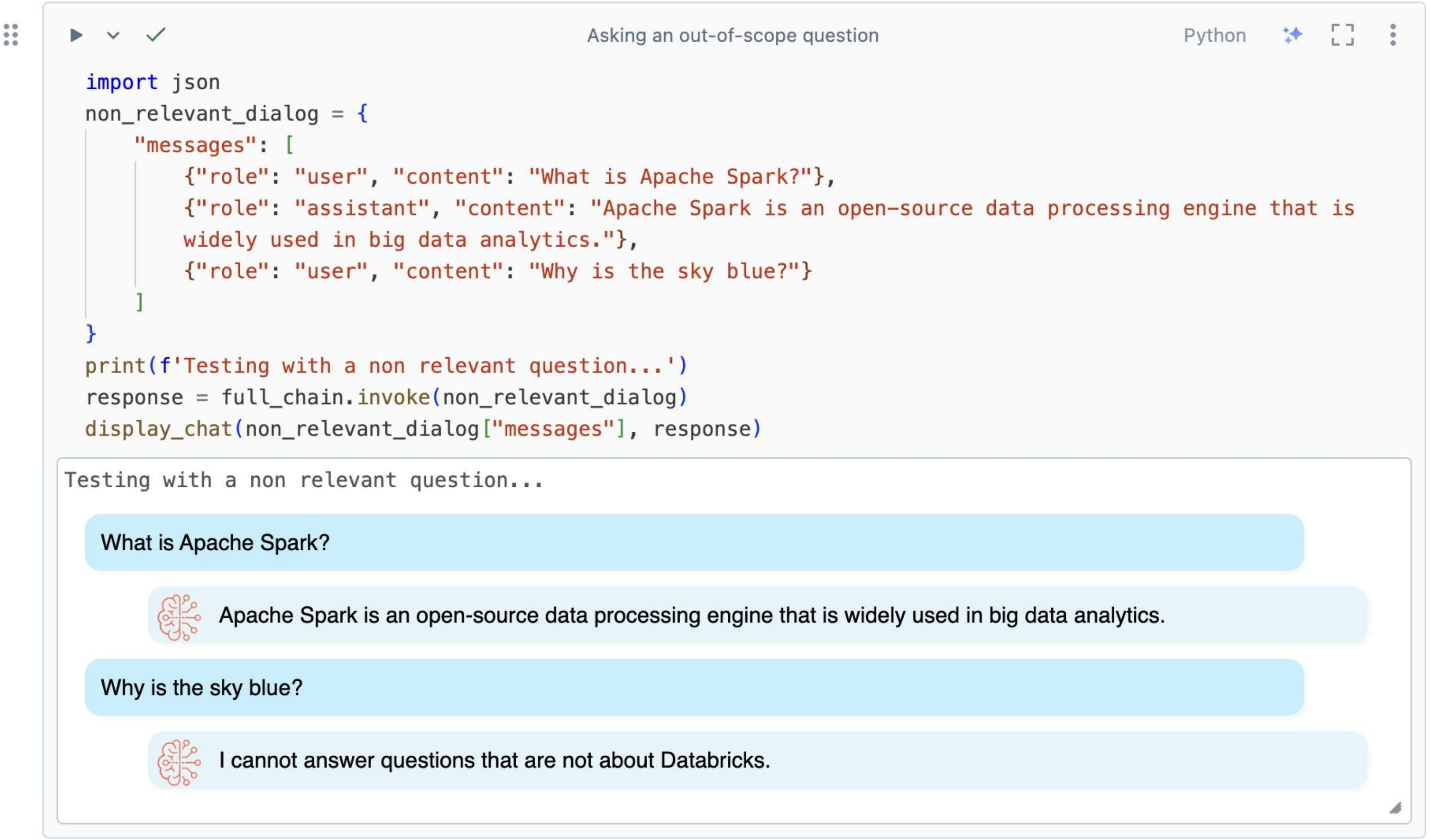
Engineer custom prompt with extended context

Knowing this context :
- *<Similar doc 1 from Vector Index>*
- *<Similar doc 2 from Vector Index>..*
Answer the question:
How to track Databricks billing?

Databricks LLM Chat endpoint
Foundation model as a service

Databricks foundation model (llama2-70B)

"Here is how you can easily track your billing: ..."



```python
import json
non_relevant_dialog = {
    "messages": [
        {"role": "user", "content": "What is Apache Spark?"},
        {"role": "assistant", "content": "Apache Spark is an open-source data processing engine that is widely used in big data analytics."},
        {"role": "user", "content": "Why is the sky blue?"}
    ]
}
print(f'Testing with a non relevant question...')
response = full_chain.invoke(non_relevant_dialog)
display_chat(non_relevant_dialog["messages"], response)
```

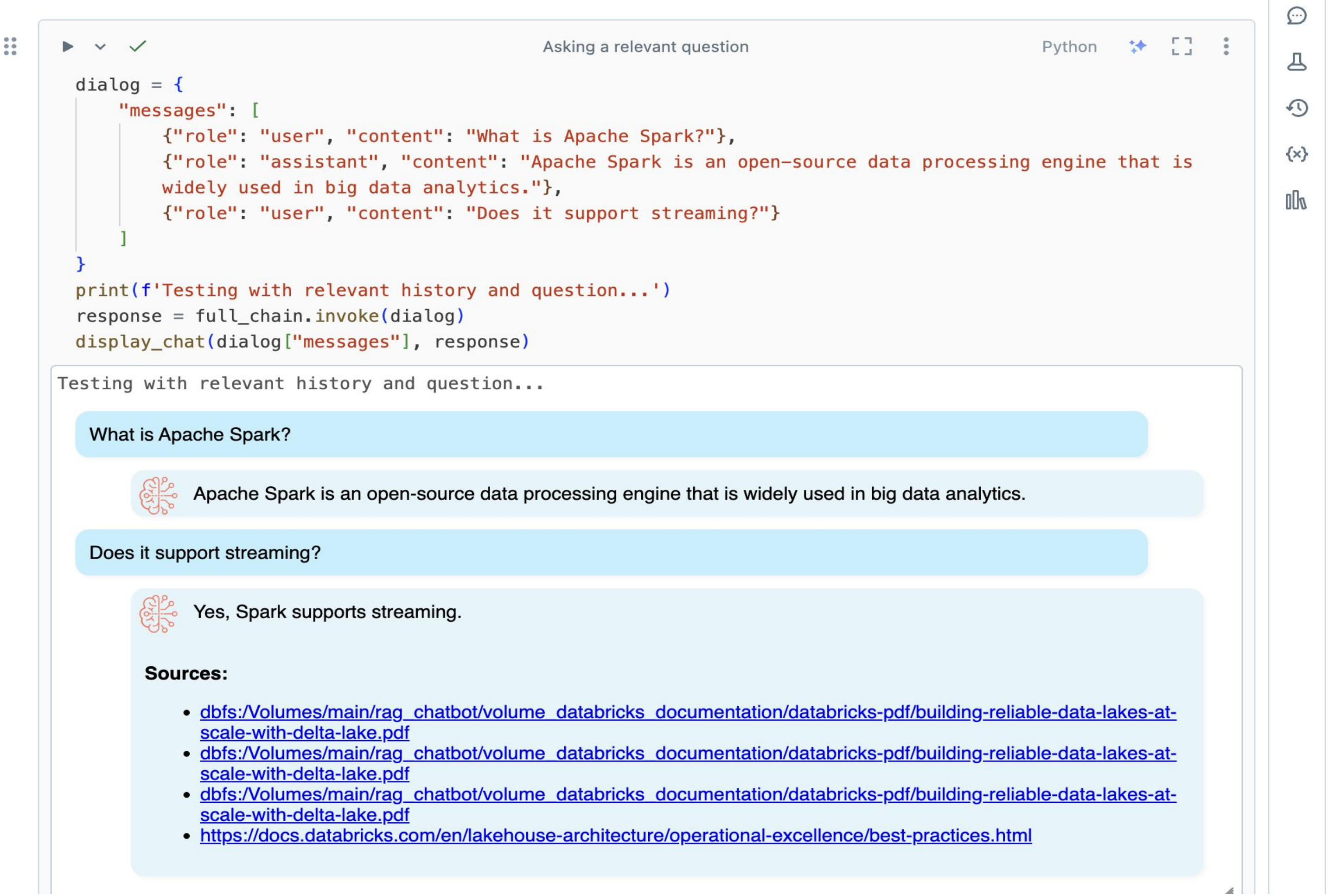
```
Testing with a non relevant question...
```

What is Apache Spark?

Apache Spark is an open-source data processing engine that is widely used in big data analytics.

Why is the sky blue?

I cannot answer questions that are not about Databricks.



```python
dialog = {
    "messages": [
        {"role": "user", "content": "What is Apache Spark?"},
        {"role": "assistant", "content": "Apache Spark is an open-source data processing engine that is
        widely used in big data analytics."},
        {"role": "user", "content": "Does it support streaming?"}
    ]
}
print(f'Testing with relevant history and question...')
response = full_chain.invoke(dialog)
display_chat(dialog["messages"], response)
```

```
Testing with relevant history and question...
```

What is Apache Spark?

Apache Spark is an open-source data processing engine that is widely used in big data analytics.

Does it support streaming?

Yes, Spark supports streaming.

Sources:

- dbfs:/Volumes/main/rag_chatbot/volume_databricks_documentation/databricks-pdf/building-reliable-data-lakes-at-scale-with-delta-lake.pdf
- dbfs:/Volumes/main/rag_chatbot/volume_databricks_documentation/databricks-pdf/building-reliable-data-lakes-at-scale-with-delta-lake.pdf
- dbfs:/Volumes/main/rag_chatbot/volume_databricks_documentation/databricks-pdf/building-reliable-data-lakes-at-scale-with-delta-lake.pdf
- https://docs.databricks.com/en/lakehouse-architecture/operational-excellence/best-practices.html

Overcoming shortcomings of Text to speech models

Unfortunately there wasn’t a dataset that we could find

{text:labels, speech:feature} for Kurdish language.

One way is to create a dataset, to reverse engineer the speech to text, make a dataset using speech from online resources and get the text. After that change the roles of feature and target so that we end up with text as the features and speech as the target. Finally build a model on the created dataset.

{speech:feature, text:labels} using existing model(s) —>

{text:labels, speech:feature} created dataset, build a model using this dataset.

All the code and resources can be found in this repository

Github repository

Previous week	Current week	Upcoming week
● Data Collection and Preparation: Engaged in extensive data gathering from Microsoft Azure, Kaggle, YouTube, and Google Speech API to create a diverse, high-quality dataset. Addressed challenges with Afghan accent in Kaggle dataset by compiling a new dataset for linguistic accuracy. ● Model Selection and Fine-Tuning: Chose to fine-tune the Couqi TTS model for improved Persian language support. Planned to develop an integrated pipeline featuring ASR, RAG, and TTS for Hindi and Persian languages. ● Innovative Evaluation and Deployment: Implemented an automated evaluation system using the Mean Opinion Score (MOS) metric. Carefully considered deployment strategies for real-world applicability. ● Collaboration and Version Control: Maintained effective team communication through Notion and established a GitHub repository for version control. ● Dataset Development and Challenges: Assembled a 25-hour dataset with single-speaker audio, free from accent inconsistencies and enriched with emotional variance. Opted against using the Common Voice dataset due to quality concerns. Technical Advancements: Bahareh Arghavani Nobar: Enhanced the XTTS architecture for Persian by updating vocab.json with unique Persian alphabets and modifying tokenizer.py for custom tokenization rules. Devnath Reddy Motati: Curated an NLP knowledge base and explored a basic RAG system using Llama2 from Hugging Face. Investigated the implementation of basic Coqui TTS in Python. Devnath also explored and implemented a basic RAG system using Llama2 from hugging face to understand the working and implementation technique.	● Project Focus: Bahareh aimed to fine-tune the XTTS v2 model for Persian audiobooks, encountering challenges and insights leading to a project pivot. ● Initial Challenges: Encountered a significant issue with DataLoader not recognizing the dataset, suggesting improper configuration. ● Research and Solution: After consulting a Medium article on XTTS v2, Bahareh adjusted the dataset to meet specific requirements (3-6 seconds audio chunks, 24KHz sample rate, mono-channel) for compatibility with XTTS and Google's Speech API. ● Training Issues: Training with a fantasy-themed audiobook revealed compatibility issues due to the XTTS model's difficulty with emotional content, resulting in distorted audio. ● Strategy Shift: Pivoted to a mono-speaker newsreader audio dataset to match the XTTS model's preference for neutral, consistent speech patterns. Expanded the model's vocab.json for Persian tokenization using Hugging Face resources. ● New Training Approach: Adopted a strategy of training the XTTS v2 model from scratch with the LJSpeech dataset, then fine-tuning on a Persian News dataset, and planning expansion to Kurdish and Hindi datasets to accommodate linguistic and acoustic diversity. ● Devnath went through the documentation available in azure regarding implementation of the RAG system using Azure Databricks. ● Completed the Collection of NLP Subject Material for Knowledge Base.	● Bahareh: Next week start training from scratch , LJspeech → Persian News→Kurdish→Hindi ● Haward: create Kurdish TTS dataset and if he find any available modify required changes on xtts to fine tune on Kurdish ● Devnath plans to deploy our model as an endpoint to be able to send real-time queries. ● Also start the process of Integrating all the system components.