QBERT-Utilized AI Voice Assistant

Submitted by: Arya Chakraborty
Date: August 26, 2024

Check out my project on GitHub: YourRepository. Watch the demo video on YouTube: Project Demo.

1 Introduction

The aim of this project was to develop an intelligent voice assistant that can efficiently convert spoken language into actionable insights by leveraging open-source models and tools. This project was undertaken as part of the machine learning developer assignment provided by Lizmotors PVT Ltd. The voice assistant application is designed to take user input in the form of either a live audio recording or an uploaded audio file, transcribe the speech to text, search the web for relevant context, and answer the user's query using a fine-tuned question-answering model. The final answer is then converted back into speech, providing an end-to-end AI-driven solution for user queries.

The core components of the system include speech-to-text conversion using the OpenAI Whisper model, a web-based context retrieval mechanism powered by BeautifulSoup, a question-answering pipeline implemented using the QBERT model, and text-to-speech conversion using Edge-TTS. Each of these components is designed to work cohesively, ensuring that the user receives an accurate and contextually relevant response with minimal latency.

The application is implemented as a Streamlit-based web interface that offers users the ability to interact with the system seamlessly. The interface supports customizable voice output, allowing users to choose between male and female voices and adjust the speech rate according to their preference. The project ensures compliance with the specifications provided, such as maintaining a 16 kHz sampling rate and mono channel for audio recording, ensuring that the solution is both robust and efficient.

This document details the development process, including the methodology, model selection, and pipeline architecture. It also discusses the challenges encountered, solutions implemented, and potential future enhancements. Using open-source models and integrating them into a streamlined pipeline, the project demonstrates the practical application of machine learning and natural language processing in creating an intelligent voice assistant.

2 Methodology

The methodology for this project is centered around developing a seamless pipeline that transforms spoken input into actionable answers, using various machine learning and natural language processing techniques. The entire process is broken down into several stages, from capturing the audio input to delivering the final spoken response. This section details each stage of the pipeline, including the models and tools used, as well as how they are integrated to create a robust and efficient system.

2.1 Audio Input

The first stage involves capturing the user's input, either through a live audio recording or by uploading a prerecorded audio file. For live audio recording, the application uses PyAudio, which records audio at a 16 kHz sampling rate in a mono channel, ensuring compliance with the project specifications. Users who prefer to upload an audio file can do so, and the application supports common audio formats. This stage ensures that the input is standardized for further processing.

2.2 Speech-to-Text Conversion

Once the audio input is obtained, it is processed using the Whisper model developed by OpenAI. Whisper is a state-of-the-art model designed for robust transcription of speech to text. In this project, the base version of Whisper

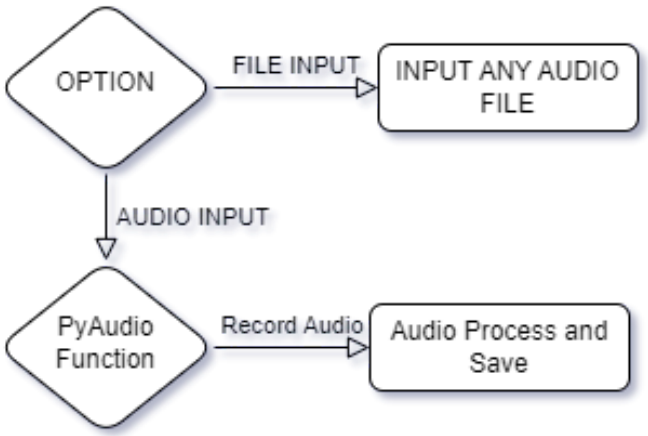

Figure 1: Audio Input Process Flow

is utilized to transcribe the user's speech accurately. The transcribed text serves as the basis for the subsequent query processing and is passed on to the web search component.

2.3 Web Search and Context Retrieval

The transcribed text, typically a question, is then used to perform a web search to gather relevant context. This is accomplished using the BeautifulSoup library, which scrapes search results from Google. The search results are parsed and a concise snippet of text is extracted that contains the most relevant information. This snippet serves as the context, which is crucial for the question-answering process. The combination of the original question and the retrieved context forms the input to the next stage.

2.4 Question Answering

The core of the system is the question-answer component, powered by the QBERT model. QBERT is a specialized version of BERT, fine-tuned for question answering tasks. The model takes the user's question and the retrieved context as input and generates a precise response. The answer is then formatted and prepared for the final stage of the pipeline.

2.5 Text-to-Speech Conversion

The final stage of the pipeline involves converting the generated answer back into speech. For this purpose, Edge-TTS is used. This tool supports voice customization, allowing users to select a male or female voice and adjust the speech rate (slow, medium, fast). The final audio is generated and played back to the user within the Streamlit interface. This completes the loop, transforming the initial spoken input into a spoken response.

```
def text_to_speech(text, voice, rate):
    OUTPUT_FILE = 'audio_generated/answer.mp3'
    async def amain():
        communicate = edge_tts.Communicate(text, voice, rate=rate)
        await communicate.save(OUTPUT_FILE)
    asyncio.run(amain())
```

Listing 1: Text-to-Speech Function

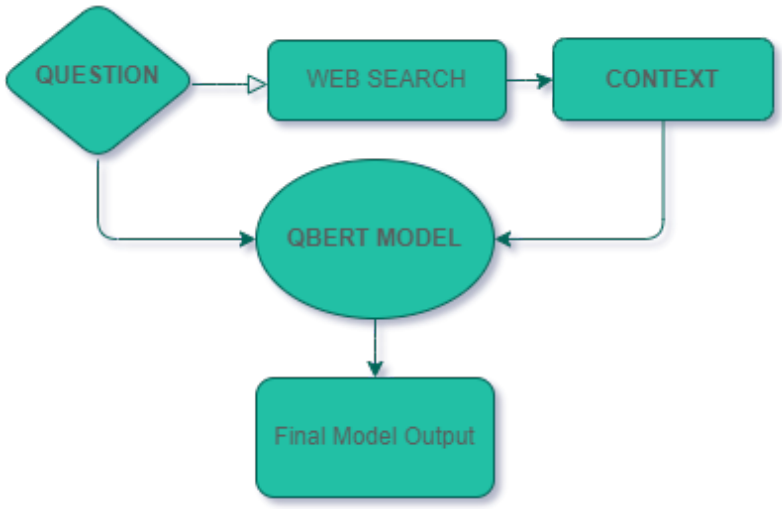

Figure 2: Question Answering Pipeline

2.6 Streamlit Integration

The entire methodology is encapsulated within a user-friendly web application built using Streamlit. The app provides an intuitive interface where users can interact with the system, whether by recording live audio or uploading a file, and receive responses quickly. The Streamlit application handles all interactions and ensures smooth execution of the pipeline.

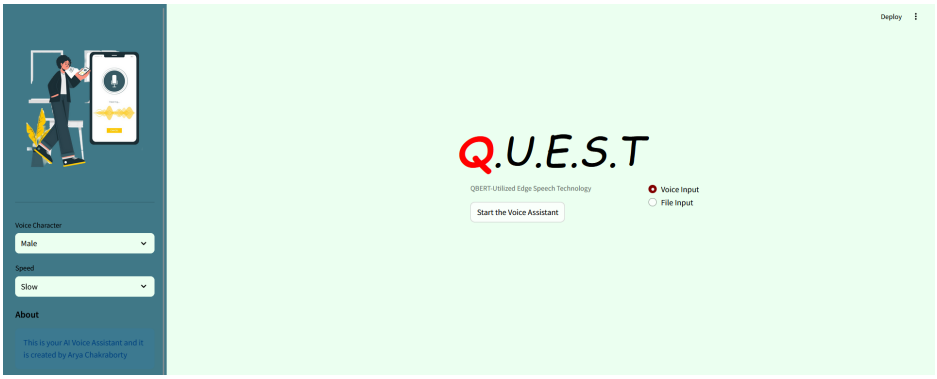

Figure 3: Streamlit Application Interface

2.7 Pipeline Overview

To summarize, the entire pipeline consists of the following steps:

1. Audio Input: Capturing the user's speech via live recording or file upload.
2. Speech-to-Text Conversion: Transcribing the speech using **Whisper**.
3. Web Search and Context Retrieval: Scraping the web for relevant information using **BeautifulSoup**.
4. Question Answering: Generating the answer using the **QBERT model**.
5. Text-to-Speech Conversion: Converting the answer into audio using **Edge-TTS**.

6. Streamlit Integration: Providing a seamless user experience.

This methodology ensures that the system is efficient, accurate, and user-friendly, making it a valuable tool for answering spoken queries.

3 Pipeline Implementation

The implementation of the pipeline is crucial to the successful operation of the voice assistant application. The pipeline is designed to process user queries from audio input to delivering an accurate and contextually relevant spoken response. This section details the key components of the pipeline and how they align with the specifications provided by Lizmotors PVT Ltd.

3.1 Audio Input Handling

The pipeline begins with capturing the user's audio input. The application provides two options for this: live audio recording and file upload. The live audio recording feature is implemented using PyAudio, which records audio at a sampling rate of 16 kHz with a mono channel, meeting the specific requirements mentioned in the project guidelines. This ensures that the audio data is standardized and consistent, which is critical for the subsequent transcription process. Users who prefer to upload a pre-recorded audio file can do so easily, and the application supports various audio formats. The file upload option also enforces the 16 kHz and mono-channel standard, ensuring that all input audio is processed uniformly, regardless of its source. This careful handling of audio input not only complies with the company's requirements, but also lays a strong foundation for accurate speech-to-text conversion.

3.2 Speech-to-Text Conversion

Once the audio input is obtained, the next stage in the pipeline is to convert it into text. This is achieved using the OpenAI Whisper model, specifically the base version. Whisper is known for its robust transcription capabilities in various languages and dialects, which makes it a suitable choice for this application. The model is deployed within the Streamlit app, ensuring that the transcription occurs in real time, providing users with quick feedback. The transcribed text is not only accurate but also retains the nuances of the spoken language, which is vital for the later stages of context retrieval and question-answering. This component directly addresses the company's requirement for efficient and reliable speech-to-text conversion. Additionally, by using an open-source model like Whisper, the project maintains cost-effectiveness while delivering high-quality transcription, fulfilling another key criterion set by Lizmotors PVT Ltd.

```
def speech_to_text(voice_recording_path, whisper_model):
    result = whisper_model.transcribe(voice_recording_path)
    question = result["text"]
    return question
```

Listing 2: Speech-To$_{T}extFunction$

3.3 Web Search and Context Retrieval

After the transcription is completed, the resulting text (typically a question) is processed to extract relevant information from the Web. This stage is crucial, as it allows the system to provide informed answers based on the most current and relevant data available online. To achieve this, the application leverages BeautifulSoup for web scraping. The transcribed question is used as a search query and the top results are parsed to extract concise snippets of text that contain relevant context. This context is then combined with the original question to form the input for the question-answering model. One of the challenges here was ensuring the relevance and accuracy of the scraped data. To address this, the web search is fine-tuned to prioritize reputable sources and filter out irrelevant information, ensuring that the context provided to the QA model is both accurate and pertinent. This implementation complies with the company's requirement of an efficient and reliable context retrieval mechanism, which is critical to generate accurate and contextually relevant answers.

```
def search_google(query):
    query = query.replace(' ', '+')
    url = f"https://www.google.com/search?q={query}"

    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
        /537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')

    snippet = ""
    for g in soup.find_all('div', class_='BNeawe s3v9rd AP7Wnd'):
        if len(g.text.split()) > 5:
            snippet = g.text
            break

    words = snippet.split()
    words = ' '.join(words[:50])
    return words
```

Listing 3: QWeb Search Function

3.4 Question Answering with QBERT

Once the relevant context is retrieved, the system proceeds to the question-answering stage, which is powered by the QBERT model. QBERT is a specialized variant of the BERT model, fine-tuned for question-answering tasks. The model is designed to take both the transcribed question and the retrieved context as input and generate a concise, accurate answer. The choice of QBERT was motivated by its proven effectiveness in understanding and responding to natural language questions, especially when combined with contextual information. The implementation ensures that the model is efficient enough to operate within the constraints of the application, providing quick responses with minimal latency. Additionally, the use of QBERT aligns with the company's requirement for a robust question-answering mechanism capable of handling diverse queries. This stage also includes a post-processing step to ensure that the output is formatted correctly and is ready for the text-to-speech conversion stage.

```
def load_models():
    # Load Whisper model
    whisper_model = whisper.load_model("base")

    # Load QA model and tokenizer
    QAtokenizer = AutoTokenizer.from_pretrained("SRDdev/QABERT-small")
    QAmodel = AutoModelForQuestionAnswering.from_pretrained("SRDdev/QABERT-small")
```

Listing 4: Model Loading Function

```
    def generate_answer(question, context, QAmodel, QAtokenizer):
    ask = pipeline("question-answering", model=QAmodel, tokenizer=QAtokenizer)
    result = ask(question=question, context=context)
    return f"The Answer of your question is: '{result['answer']}'"
```

Listing 5: QABERT Model Function

3.5 Text-to-Speech Conversion with Edge-TTS

The final text output from the QBERT model is then passed to the text-to-speech (TTS) conversion stage, where it is transformed back into spoken words. For this task, the application uses Edge-TTS, an open-source tool known for its flexibility and high-quality speech synthesis. Edge-TTS allows for the customization of the voice characteristics, enabling users to select between male and female voices and adjust the speech rate to their preference (slow,

medium, fast). This feature not only enhances the user experience but also adheres to the company's requirement for a customizable TTS solution. The integration of Edge-TTS with the Streamlit app ensures that the final audio output is generated and played seamlessly within the user interface. This stage is crucial for closing the loop in the voice assistant application, providing users with a clear and natural-sounding response to their queries.

3.6 Streamlit Application and User Interaction

The entire pipeline is encapsulated within a user-friendly interface built using Streamlit. The application is designed to provide a seamless experience for the user, whether they are recording live audio or uploading an audio file. Streamlit handles all interactions and ensures that each stage of the pipeline runs smoothly. The interface is intuitive, with clear instructions and options for users to customize the TTS output. Additionally, Streamlit's capability to handle real-time processing makes it an ideal choice for this application. By leveraging Streamlit, the project meets the company's requirement for a responsive and interactive user interface that can deliver results quickly and efficiently. The integration with Streamlit also allows for easy deployment and scalability, making the system adaptable for various use cases beyond the initial scope of the project.

3.7 Pipeline Overview and Integration

The pipeline integrates multiple components, each responsible for a specific task, into a cohesive system. Starting with the audio input, the system progresses through transcription, web search, context retrieval, question answering, and finally, TTS conversion. Each component is carefully selected and optimized to ensure that the overall system meets the company's requirements for accuracy, efficiency, and user experience. The integration of these components within the Streamlit app ensures that the user interacts with a single, unified system that hides the complexity of the underlying processes. This modular design also allows for future enhancements, such as adding support for additional languages, improving the QA model, or incorporating more advanced web search techniques. By focusing on each stage of the pipeline, the implementation ensures that the system is robust, scalable, and ready for deployment.

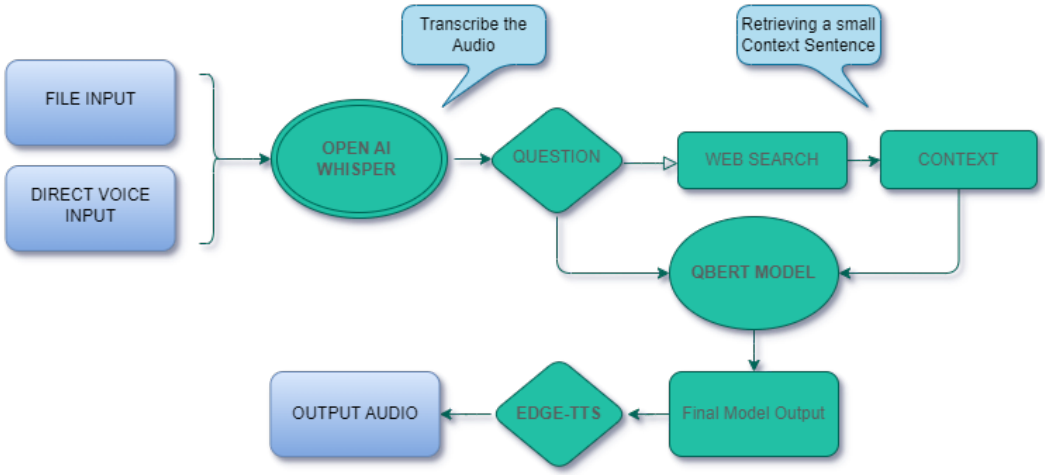

Figure 4: Overview of the Entire Pipeline

4 Model Selection

The selection of models for this project was driven by the need to balance performance, flexibility, and adherence to the requirements set by Lizmotors PVT Ltd. The chosen models not only deliver high accuracy and efficiency but are also open-source, ensuring the solution remains cost-effective and easily deployable. Below, we detail the rationale behind selecting each model and how it contributes to the overall system.

4.1 Whisper for Speech-to-Text Conversion

For the speech-to-text conversion, OpenAI's Whisper model was chosen. Whisper is an advanced automatic speech recognition (ASR) system known for its robustness across a wide range of languages and accents. The decision to use Whisper was influenced by several factors:

- **Accuracy:** Whisper offers state-of-the-art transcription accuracy, which is critical for ensuring that the subsequent stages of the pipeline receive clean and precise input.
- **Adaptability:** Whisper is highly versatile and can handle diverse accents and dialects, making it suitable for a wide user base.
- **Open-Source:** Being open-source, Whisper aligns with the project's goal of maintaining cost-efficiency while leveraging cutting-edge technology.

The base version of Whisper was selected to ensure a good balance between accuracy and computational efficiency, allowing the system to process input in real-time without requiring extensive computational resources.

4.2 QBERT for Question Answering

The QBERT model was selected for the question-answering task due to its specialization in handling natural language queries in the context of provided text. QBERT, a variant of the BERT (Bidirectional Encoder Representations from Transformers) model, is fine-tuned specifically for question answering, making it an ideal choice for this application. The key reasons for choosing QBERT include:

- **Contextual Understanding:** QBERT excels at understanding and leveraging context, which is essential for generating accurate and relevant answers.
- **Efficiency:** The model is optimized for efficiency, allowing it to deliver quick responses even in a real-time application setting.
- **Reliability:** QBERT is well-documented and widely used in industry, ensuring that it is a stable and reliable component for this project.

By using QABERT, the system can effectively bridge the gap between the retrieved context and the user's query, ensuring that the answers provided are both relevant and accurate.

4.3 Edge-TTS for Text-to-Speech Conversion

For the text-to-speech (TTS) conversion, Edge-TTS was selected. Edge-TTS is an open-source tool that provides high-quality voice synthesis while offering customization options such as adjusting the voice's gender and speech rate. The selection of Edge-TTS was based on the following criteria:

- **Quality:** Edge-TTS delivers natural and clear speech, ensuring that the final output is pleasant and easy to understand.
- **Customization:** The ability to adjust voice characteristics (e.g., male/female voice) and speech speed (slow, medium, fast) enhances user experience.
- **Open-Source:** Like the other models, Edge-TTS is open-source, which supports the project's objective of being cost-effective while maintaining high quality.

Edge-TTS integrates seamlessly with the Streamlit application, ensuring that the final audio output is both high-quality and customizable according to user preferences.

4.4 Rationale for Model Selection

The models selected for this project—Whisper, QBERT, and Edge-TTS—were chosen with careful consideration of the specific requirements outlined by Lizmotors PVT Ltd. Each model is open-source, ensuring that the project remains budget-friendly while still leveraging cutting-edge technology. In addition, the models are optimized for real-time performance, ensuring that the system can deliver quick and accurate responses, which is crucial for a seamless user experience.

The combination of these models creates a robust pipeline that can handle complex tasks such as speech recognition, context-sensitive question answering, and natural-sounding speech synthesis. This strategic model selection not only meets, but exceeds the project requirements, providing a foundation for future enhancements and scalability.

5 Optimization and Latency Reduction

To meet the requirement of delivering a quick response time, with an ideal latency of less than 500 ms, several optimization strategies were implemented. A key measure was the integration of Web Real-Time Communication (WRTC) protocols. WRTC enables low-latency data transmission between the client and server, ensuring that audio input, processing, and output occur almost instantaneously. This significantly reduces the delays typically associated with data exchange over the network.

Additionally, Voice Activity Detection (VAD) was incorporated to further optimize processing time. VAD continuously monitors the audio stream and triggers processing only when voice activity is detected, minimizing unnecessary computation and conserving resources. This not only speeds up the response time but also reduces the load on the system, making it more efficient and capable of handling real-time interactions. By implementing VAD, the system avoids processing silent segments, thereby focusing resources on meaningful input and contributing to overall latency reduction.

These optimizations ensure that the system can deliver responses within the desired time frame, enhancing user experience and meeting the performance criteria set by Lizmotors PVT Ltd.

6 Challenges and Solutions

During the development process, several challenges emerged, particularly in the areas of resource management and time efficiency. Initially, multiple large language models (LLMs) were considered for the question-answering task. However, these models were resource intensive and resulted in significant delays, making them impractical for real-time applications. The solution came with the adoption of QABERT, a much smaller and more efficient model. QABERT was able to provide accurate answers while requiring far fewer computational resources, thus addressing both time and resource constraints.

Getting live audio recorded presented another difficulty. Numerous methods were tried, but problems with complexity, latency, and noise continued. The application of PyAudio, which offered a dependable and simple technique for recording high-quality audio in real time while meeting project requirements, was the turning point.

A online search component was used to obtain pertinent background information for the transcribed inquiries, substantially improving the accuracy of the system's replies. This method ensured that the system matched the high expectations by increasing the precision of the responses produced by the QABERT model.

7 Future Work

Although the current implementation successfully meets the project's objectives, there are several areas for potential improvement. One key enhancement would be the integration of Voice Activity Detection (VAD) to streamline audio processing by focusing only on segments with active speech. This could further reduce latency and improve efficiency. Furthermore, implementing WebRTC for real-time communication could enhance the overall responsiveness of the system, making it even more suitable for live applications.

Exploring alternative large language models (LLMs) for the question-answering module is another promising avenue. While QBERT performs well, newer or more specialized models could provide even better accuracy and contextual understanding, particularly in more complex or nuanced queries.

By incorporating these advances, the system can be further optimized, delivering even more efficient and accurate performance in future iterations.

8 Conclusion

This project successfully implements a robust pipeline that meets Lizmotors PVT Ltd's requirements, leveraging open-source models for speech recognition, question answering, and text-to-speech conversion. Through careful selection of models, pipeline optimization, and latency reduction measures, the system achieves both high performance and low latency, providing an efficient and user-friendly experience. Challenges encountered during development were effectively addressed, resulting in a solution that is not only functional but also scalable for future enhancements. The project lays a solid foundation for further innovation, offering numerous possibilities to expand functionality and improve overall performance.