Nova

Hamid Nazari
Yosafe Feseha Oqbamecail
Adrian Hatlenes Blikås
Khartira Arif

Date: 21.04.2024

Introduction

This project presents a web-based virtual assistant with voice activation capabilities. The assistant leverages OpenAI's API for generative text responses and ElevenLabs for converting text to speech, providing a user-friendly and interactive experience. We opted for this approach after evaluating and experimenting with various speech-to-text APIs like Google's offering, CMU Sphinx, Mozilla DeepSpeech, and Whisper.

Our motivation for choosing this project stemmed from the growing interest for voice-enabled interfaces and the desire to explore the capabilities of generative AI models in creating highly engaging and intuitive user experiences. With the rapid advancements in natural language processing (NLP) and speech synthesis technologies, the potential for developing virtual assistants that can understand and communicate with users in a more human-like manner has become a reality.

Our project isn't alone in exploring the potential of voice-activated virtual assistants powered by generative AI. Major tech companies are actively trying to develop similar products. Apple, for instance, is exploring the possibility of replacing Siri, its current AI assistant on iPhones, with one powered by generative AI. Similarly, OpenAI itself has ambitions to create an advanced voice-activated virtual assistant and offer it as a service to other businesses.

Background

Generative AI models have emerged as a transformative force in the field of computer science, revolutionizing the way we create and generate content across various domains. These models, powered by deep learning techniques, are essentially algorithms trained on massive datasets. This training allows them to learn and understand the underlying patterns and structures within the data, be it text, images, or audio. Once trained, generative AI models can use this knowledge to produce entirely new and coherent content that mimics the characteristics of the data they were trained on.
For instance, a generative model trained on a vast corpus of news articles can generate realistic and grammatically correct news stories. Similarly, models trained on image datasets can create original photographs or even modify existing ones. The impact of generative AI extends beyond mere content creation.

In the realm of speech synthesis, generative AI models have played an important role in overcoming the limitations of traditional text-to-speech (TTS) systems. Traditional TTS systems often relied on rule-based approaches, resulting in robotic or monotonous outputs that lacked the natural inflections and nuances of human speech.

However, with the advent of generative AI models, like those employed by ElevenLabs, speech synthesis has undergone a significant transformation. These models are trained on vast amounts of audio data, allowing them to capture the intricacies of human speech patterns, including tone variations, emotional intonations, and natural pauses. This newfound ability to generate highly realistic and natural-sounding speech has opened doors for a variety of applications.

Methodology

The backend of our application is built using Python and the framework FastAPI. This framework simplifies the development process for building APIs and facilitates the integration of OpenAI's API for text generation functionalities and function calling. ElevenLabs' API is used on the backend to seamlessly convert the generated text responses from OpenAI’s API into natural-sounding speech for user output.

The reason we chose FastAPI over other alternatives is because FastAPI offers several advantages for building our application's backend. It is high-performance, making it ideal for real-time interactions with the virtual assistant. Additionally, FastAPI's automatic documentation generation simplifies deployment and future maintenance.

The frontend is developed using React, a popular JavaScript library for building user interfaces, and Tailwind CSS, a utility-first CSS framework that allows for rapid and efficient styling of the application. The frontend is responsible for capturing the user's voice input, sending it to the backend for processing, and playing the generated speech response. The voice is captured with the help of a React library called *React Media Recorder*, which is a React component library that allows for easy integration of media recording functionality into React applications. This library simplifies the process of capturing audio and video from a user's device using the MediaRecorder API, which is supported in most modern web browsers.

Even though there are many other alternatives, we chose OpenAI's API for text generation. The reason we went with this option is because OpenAI's API provides state-of-the-art natural language processing capabilities, including text generation and understanding. Another important reason why we chose OpenAI is due to its functional calling feature. This unique feature allows developers to create a more interactive experience. Through this feature, we can instruct the model to perform specific actions that are beyond just text generation. .By interfacing with OpenAI's API, our application can accurately comprehend user commands and generate relevant responses.

In addition to using OpenAI's API regular text generation, we have also used it for converting speech into text . When the backend of our application receives audio files from the frontend part, the audio files are then converted to text with help of OpenAI's text-to-speech API.

ElevenLabs API, a specialized text-to-speech solution, was selected to deliver the virtual assistant's responses in a natural and engaging manner. Unlike generic text-to-speech engines, ElevenLabs uses generative AI models that are trained on extensive speech datasets. This allows the model to capture the subtleties and nuances of human speech patterns, which results in highly realistic and natural-sounding voice outputs.

By integrating ElevenLabs, our application ensures that user interactions feel more natural and engaging. The listener perceives the responses as coming from a human speaker rather than a robotic voice, enhancing the overall user experience.

Application Design and Development

The application follows a client-server architecture, where the backend, built with FastAPI and Python, handles the core logic of the application. It integrates with the OpenAI API for natural language processing and understanding, interpreting the user's

voice input and determining the appropriate response. Once the response is generated, the backend uses the ElevenLabs API to synthesize natural-sounding speech output, which is then sent back to the frontend for playback.

The design of the user interface took inspiration from some popular designs we found on Dribbble, which is a self-promotion and social networking platform for digital designers.

Experiments and Results

During the development process, we conducted various experiments and tests to evaluate the performance and accuracy of the chosen technologies. These experiments involved testing the application with a diverse set of voice commands and system prompts, ranging from simple queries to more complex and contextual requests.

To enable speech synthesis capabilities, as briefly mentioned in the introduction, we explored many different alternatives, including CMU Sphinx (PocketSphinx), Mozilla DeepSpeech, and Whisper, in addition to ElevenLabs.We tested and evaluated each of these options, and then compared them in order to choose the best one.

After the above mentioned experiments, we found that ElevenLabs outperformed the other alternatives in generating highly realistic and natural-sounding speech outputs. While some solutions like CMU Sphinx and Mozilla DeepSpeech offered good accuracy and resource efficiency, they fell short in producing truly natural-sounding speech, often resulting in robotic or monotonous outputs.

Challenges and Problem-Solving

One of the main challenges we encountered during the project was the integration of the different APIs and ensuring seamless communication between the various components, and choosing the right set of technologies. Each API had its own set of requirements, protocols, and data formats, which needed a lot of research and experimentation to understand and properly integrate.

Another significant challenge that we faced in the initial phase of the project involved selecting the most suitable speech-to-text service. We spent a lot of time researching and exploring various options and alternatives, considering factors such as accuracy, real-time processing capabilities, natural-sounding output, cost-effectiveness, and

customizability. We researched and explored services, including Google's Speech-to-Text API, CMU Sphinx (PocketSphinx), Mozilla DeepSpeech, and Whisper.

After careful consideration and testing, we ultimately chose the service offered by ElevenLabs. We found out that ElevenLabs' text-to-speech solution is the best in the entire industry in terms of generating highly realistic and natural-sounding speech outputs, while also offering a good balance of accuracy, real-time processing, and customizability.

Discussion

When we initially decided to choose this project, we discussed building and training our own models from scratch, both for text generation and speech synthesis. However, after further research and analysis, it became clear to us that building custom models would require significant computing power and access to vast amounts of data, which was not feasible within the scope and resources of this project.

Building and training large language models and speech synthesis models from scratch is a complex and resource-intensive task, often requiring specialized hardware (such as GPUs or TPUs) and access to massive datasets. Additionally, the training process itself can take weeks or even months, depending on the complexity of the models and the size of the datasets.

Given these limitations, we decided to use existing pre-trained models and APIs provided by industry leaders, such as OpenAI and ElevenLabs. These services offer robust and well-trained models that have been developed and optimized by teams of experts, allowing the project to benefit from their cutting-edge technology without the need for extensive model training and infrastructure.

Reproducibility

To ensure the reproducibility of this project, we have used a version control system and created a GitHub repository.The project's GitHub repository is well-organized and contains clear instructions for setting up and running the application, as well as any necessary dependencies and configurations.

Additionally, the application's codebase is modular and follows the principles of separation of concerns, making it easier for others to understand, maintain, and extend the codebase in the future.

Conclusion

The voice-activated virtual assistant project demonstrates the potential of generative AI models in creating engaging and natural-sounding voice interfaces. By using modern technologies, this application provides users with a seamless and intuitive experience, enabling them to interact with the virtual assistant through voice commands in a more natural and human-like manner.

During the development of this project, we learned a great deal about different technologies and the possibilities that can be achieved with modern deep learning techniques. The process of integrating various technologies and external APIs to build a web-based voice-activated virtual assistant has been a very rewarding and educational experience for all of us. In conclusion, we have successfully integrated various technologies and external APIs to build a fairly good web-based voice-activated virtual assistant.

References

- Inside Apple's Big Plan to Bring Generative AI to All Its Devices.
- OpenAI Wants to Build a Personal Assistant for the AI Age.
- OpenAI API Documentation.
- ElevenLabs API Documentation.
- FastAPI Documentation.
- React Documentation.
- Tailwind CSS Documentation.