Web App: Voice-To-Text with Grammar and Spelling Correction

Yanyan Li, Hongxin Wu, Xinran Zhang

Table of contents

Introduction

With the rise of globalization, an increasing number of young people are going abroad for cultural and educational experiences. However, non-native speakers often encounter challenges in understanding a new language, including accents, speech speed, and more. For instance, in China, dialects vary greatly; someone from the north might struggle to comprehend Mandarin spoken in the south. Our personal experience in a classroom with similar challenges inspired this project. It led us to wonder: Is there lightweight speech recognition software that can assist us in understanding complex pronunciations, swallowed words, and rapid speech?

Most existing software can only perform basic speech recognition. However, for non-native speakers, it's crucial to process various dialects and unclear articulations, adding grammar

and spelling corrections to the translated text. This gap in the market sparked the initiation of our project.

Selecting suitable data was a significant challenge. Using overly clear audio clips for training wasn't ideal, as they didn't represent the real challenges faced by non-native English speakers like us. We chose recordings from online university classes and Ted speaking, which features professors worldwide with diverse accents. If our model could learn to interpret these various speech patterns, it might alleviate our confusion and bring us closer to our objective.

In choosing the right models, we explored pre-trained options on Hugging Face to meet our quality and design goals. Our aim was accuracy in a lightweight package, so we opted for models with fewer parameters and quick response times. We selected `facebook/wav2vec2-base-960h` for speech recognition to text translation, `vennify/t5-base-grammar-correction` for grammar correction, and `oliverguhr/spelling-correction-English-base` for spelling correction.

Beyond developing this voice-to-text software with integrated grammar and spelling correction, we've taken an extra step by creating an interactive web application. This user-friendly app allows users to record and playback their voice easily. It's accessible for users of all levels and includes the capability to transcribe recordings, displaying both original and corrected transcripts. This feature addresses grammatical and spelling errors. A key advantage of our web application is its fast response time and the ease of controlling recording durations. This gives international students, especially those whose first language isn't English, a valuable tool to better understand their lectures.

Through this project, we have extended the boundaries of traditional voice-to-text transcription and correction by introducing a tool that is not only theoretically sound but also immensely practical. It is designed to improve the learning experience for students worldwide, enhancing their ability to understand and communicate across a variety of dialects and linguistic challenges in educational settings.

Data preparation

To prepare our model for training, we first needed to gather a large amount of data, containing both auditory and textual elements. Our strategy involved looking for public speeches from online platforms such as YouTube and various educational websites. This approach allowed us to obtain both the spoken version of these speeches as well as their corresponding written transcripts.

Once we had collected these resources, the next step was to integrate the audio files with their text counterparts to construct a unified training dataset. An important part of this process involved cleaning and standardizing the text data. We aimed to remove any extraneous formatting that might interfere with the model's learning process.

Finally, we converted the speech audio from the downloaded YouTube content into a structured list of sentences. This comprehensive list, along with the refined text data, formed the foundation of our dataset, which provided us with the necessary materials for training our model.

Modeling

In our project, we aimed to develop a comprehensive system capable of accurately transcribing and refining speech. This task could be split into three steps: initially converting voice recordings to text, followed by correcting any spelling errors, and finally polishing the text for grammatical accuracy. Our approach was to individually select three specialized models for each of these tasks and then integrate them to form an effective, unified system. To enhance their performance, we incorporated techniques like fine-tuning and hyperparameter tuning.

We selected three models popular for their capabilities in their respective areas: `facebook/wav2vec2-base-960h` for voice-to-text conversion, `oliverguhr/spelling-correction-english-bas` for spelling correction, and `vennify/t5-base-grammar-correction for grammar` improvement. These models are all available on the Hugging Face platform.

The first model, `facebook/wav2vec2-base-960h`, is a key component of the ASR (Automatic Speech Recognition) system. It's primarily used for converting spoken language into text. We chose this model for its efficiency in self-supervised learning and its high accuracy in transcribing speech.

Following the transcription, the `oliverguhr/spelling-correction-english-base` model comes into play. Its role is to refine the transcribed text and correct any spelling errors. This step is crucial to ensure the text's accuracy and clarity. The model's ability to identify and correct a wide range of spelling mistakes efficiently made it an ideal choice for our system.

The final step involves the `vennify/t5-base-grammar-correction` model, which addresses grammatical errors in the text. The T5 model's advanced understanding of language context enables it to execute comprehensive grammatical adjustments, thus enhancing the overall readability of the text.

By integrating these models, our system not only transcribes spoken language with precision but also ensures that the text output is grammatically sound and free from spelling errors. The process involves a sequential flow, where each stage of correction builds upon the previous, resulting in high-quality, reliable textual output.

Moreover, we also optimize these models through the fine-tuning phase, which adjusts the pre-trained models to better fit our specific dataset. We also have our model hyperparameter tuning, where we experiment with different settings of the models' parameters. This helps us figure out the best configuration for our needs. By using all these methods, as mentioned, we

aim to improve our model's accuracy and efficiency in handling our unique speech and text data.

Fine Tuning and Results

Interactive Web Application

In an effort to extend the reach and impact of our voice-to-text with grammar and spelling correction model, we have developed an interactive web application. This application serves as a direct, real-time interface for the model, embodying simplicity and effectiveness. It's designed to engage users and make the sophisticated technology behind our model accessible and user-friendly.

Web Application Design and Features

Our web app features a design that emphasizes ease of use and minimalism. The interface is intuitive, focusing on functionality, which ensures that users of all levels can navigate and utilize it without complexity. Key features of the webpage include:

- **Recording** At the forefront of the app is the 'Record' button. When the user clicks on it, it will not only instantly start recording the sound by using the user's local microphone but also have a countdown timer appear in the center of the screen, reminding the user of the end of the recording time. Extending the recording time is not a difficult task; it can be easily realized by modifying the time control parameter in `app.py` and `script.js`.
- **Playback** When the user clicks the 'Play' button, the sound clip just recorded will be played back through the user's system speakers in time. This helps the user better learn and understand the original audio in conjunction with the translated text.
- **Transcription Display** Clicking the 'Transcribe' button triggers the display of the translated text in the text area at the bottom of the app. This translation is generated from the recorded audio using the `facebook/wav2vec2-base-960h` model, providing users with an immediate, native translation.
- **Correction Display** The 'Correct Text' button reveals the output of our grammar and spelling correction model in the text area. This feature aids users in better understanding what the lecturer talked about in the recorded clips.
- **Styling and Responsiveness** Our web app is styled using a `style.css` file, ensuring the interface is not only visually appealing with a light blue, 'cute' theme but also responsive to different device screens. The app is responsive to different screen sizes, ensuring it is accessible on various devices. To make our app appear more professional,

every component in our app, including its colors and positioning, has been thoughtfully designed and implemented.

Technical Components and Functionalities

The backend of the web application, developed in `app.py` handles user requests and processes voice and text through our voice-to-text grammar and spelling correction model. Using `Flask`, a web framework, `app.py` efficiently manages web requests and dynamically generates content. It acts like a command center, ensuring all user requests are accurately fulfilled.

`corrected.py` plays a crucial role in the backend. It contains the logic for our voice-to-text grammar and spelling correction model, which is then used by app.py to process user input voice and generate original, transcribed, and corrected paragraphs. Essentially, `corrected.py` takes spoken words, identifies them, and then refines them for grammar and spelling. This script is key to ensuring the text output is coherent and error-free. It works hand-in-hand with `app.py` to convert spoken language into polished, understandable text.

`script.js` is the part of our web app that makes everything work smoothly on the website. It's got code that lets users record their voice and then play it back. It also handles sending the recording to our server to get it transcribed (which means turning the audio into text). After we get the text, the script can also ask the server to fix any grammar or spelling mistakes. Basically, it's what makes our web app interactive and user-friendly, letting people record, listen, and get their spoken words turned into easy-to-read text with corrections.

The `asr.py` file is a key component of our web app's backend. It's all about handling the audio stuff. When someone records their voice on the website, asr.py takes over. As we mentioned in the prior sections, the pre-trained model could also turn that audio into transcribed text. This is super useful because it's the first step in getting our app to understand, and then it could correct what users say in the following steps. It's like the ears of our project, ensuring every spoken word is captured and transcribed accurately.

Deployment and Usage

Our web application is user-friendly and easily deployable for individuals of all skill levels. To start, simply run `python app.py` at your terminal locally. The app features an intuitive design, allowing users to record their voices with a single click. After recording, it processes the audio to transcribe and correct grammar and spelling errors. Deployment is also straightforward: download the code, set it up on your server, and it's ready for use. Our app, with its user-centric design, is an ideal tool for both students and teachers, simplifying the learning and teaching processes.

Conclusions and Future Works

conclusions

In short, our project goes beyond just theory. We've created a tool that not only aids in understanding and transcribing spoken language but also enhances it through grammar and spelling corrections. It has taught us about the difficulties and complexities of speech recognition, especially in dealing with accents and dialects. In future iterations, there's still room for us to improve. We could expand our dataset to include a wider variety of accents and colloquial speech that the model fails to pick up on and further tune it to improve its accuracy. On the application front, we aim to integrate live correction features, offering users immediate feedback and suggestions, much like what's seen in advanced writing aids. This evolution will further our goal of breaking down language barriers and making communication more accessible to non-native speakers.