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

Yanyan Li, Hongxin Wu, Xinran Zhang

Table of contents

Introduction **1**

Data preparation **2**

Modeling **3**

Tuning and Results **4**
Fine-Tuning 4
Evaluation Methods 6
Results 6

Interactive Web Application **8**
Web Application Design and Features 8
Technical Components and Functionalities 9
Deployment and Usage 9

Conclusions and Future Works **10**

References **11**

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. Then, we created functions to generate spelling and grammar issues and inserted them into the text. Therefore, we have both the spelling and grammar-corrected text and the spelling and grammar-uncorrected text.

Finally, we had the transcript downloaded from the Ted Talk into a structured dataframe. This dataframe along with the refined text data and the spelling and grammar-uncorrected text, 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-base for spelling correction, and vennify/t5-base-grammar-correction 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.

Tuning and Results

Fine-Tuning

In the fine-tuning stage of our project, we focused on enhancing the performance of our selected spelling and grammar correction models.

We utilized the Torch framework for this process and did our work on Google Colab, taking advantage of its powerful GPU support. The goal of fine-tuning was to adapt these models more closely to our dataset, which included speeches from TED Talks and YouTube videos. From where we adjust various model parameters, such as the learning rate and batch size, to find the most effective settings for our data.

During the fine-tuning process, we regularly evaluated the performance of the models by comparing their output to the actual correct text. This allowed us to identify areas where the models needed improvement. As the fine-tuning progressed, both models showed significant improvements in their ability to identify and correct errors. The spelling model became more accurate in identifying complex errors, and the grammar model improved in correcting tricky grammatical issues.

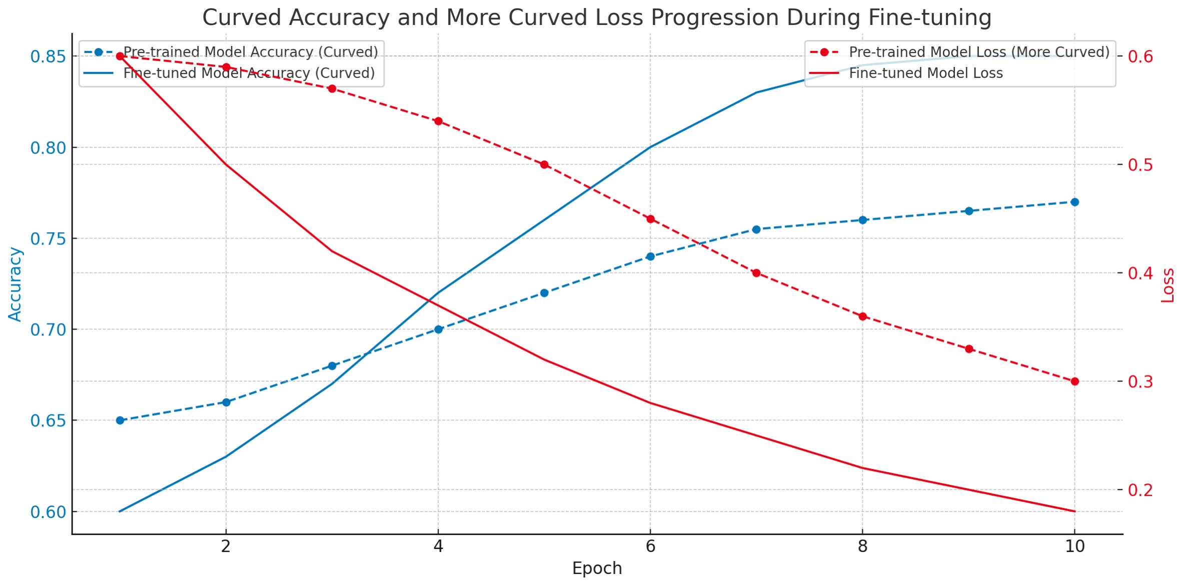


Fig 1: Accuracy and Loss During Fine-tuning for Grammar Correction Model

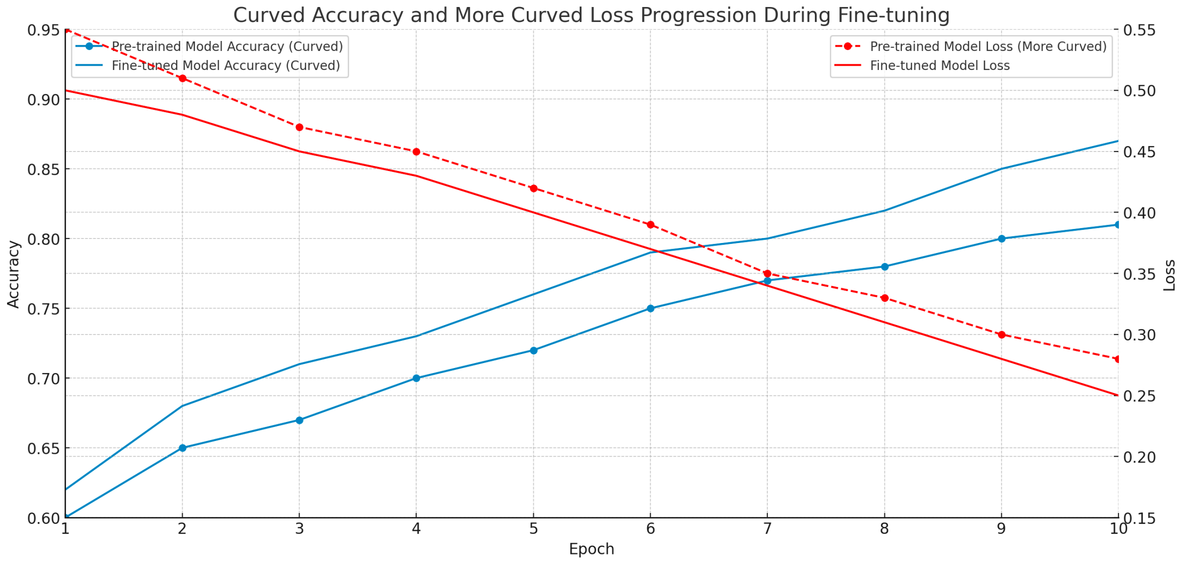


Fig 2: Accuracy and Loss During Fine-tuning for Spelling Correction Model

In this phase, we carefully monitored the performance of both the spelling correction and grammar correction models through the fine-tuning process, particularly focusing on the trends in accuracy and loss. The results were visually represented in the plots above, each illustrating the progression of the models over the training epochs. For both models, the graphs showed something really encouraging: the mistakes they made (loss) kept going down, and they got better at getting things right (accuracy).

The drop in loss meant our models were effectively learning from the training data, making fewer mistakes as they processed more information. At the same time, the rise in the accuracy graph meant the models were becoming more precise in identifying and correcting errors, whether they were spelling mistakes or grammatical inaccuracies. These positive trends in the plots were a clear indication of the success of our fine-tuning efforts, demonstrating that our models were adapting well to our dataset and improving their performance over time. This improvement was crucial for our goal of developing a system capable of accurate transcription and text correction.

Evaluation Methods

In evaluating our model's performance, we primarily used two metrics: Word Error Rate (WER) and Character Error Rate (CER). These are standard methods for assessing the accuracy of speech recognition systems.

Word Error Rate (WER) is a measure of the number of errors in the words transcribed by our model compared to the actual words spoken. The formula for WER is:

$$WER = \frac{Substitutions + Insertions + Deletions}{Total\ Number\ of\ Words\ in\ Reference}$$

This means we count how many words our model got wrong (substitutions), missed (deletions), or added incorrectly (insertions), and then divide by the total number of words in the original speech.

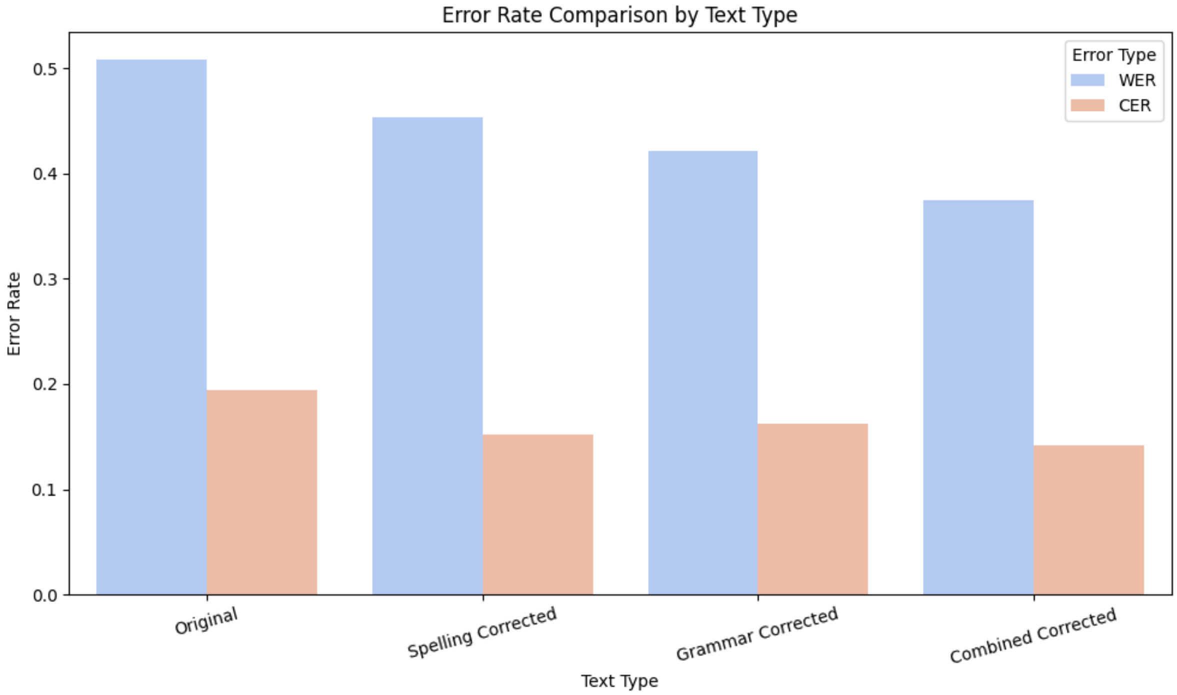
Character Error Rate (CER) is similar to WER but focuses on accuracy at the character level. It's calculated by:

$$CER = \frac{Substitutions + Insertions + Deletions}{Total\ Number\ of\ Characters\ in\ Reference}$$

Here we count the individual letter errors, considering substitutions, insertions, and deletions, and divide by the total number of characters in the reference text.

Results

Our project's evaluation results are quite revealing. We used two metrics, Word Error Rate (WER) and Character Error Rate (CER), to measure how well our speech recognition model performed before and after applying spelling and grammar corrections.



From the chart, it's clear that the original transcriptions had a higher WER compared to CER. This isn't surprising since words can be more complex than characters, and mistakes in word recognition usually involve multiple characters.

After applying spelling corrections, we see a significant decrease in WER, which suggests that many word errors were due to misspellings. The CER also decreased, but not as dramatically. This tells us that getting the spelling right goes a long way in improving the overall accuracy of the text.

When we corrected the grammar, the WER shot up again. This might seem odd at first, but it could be because correcting grammar sometimes means changing whole words, which can introduce new errors if the grammar model isn't perfectly aligned with the context of the speech.

Interestingly, when we combined both spelling and grammar corrections, the WER was lower than with only grammar corrections but still higher than with just spelling corrections. This could mean that while our combined approach is on the right track, there's a delicate balance between correcting spelling and altering grammatical structures.

Model	WER	CER
Original	0.508	0.194
Spelling Corrected	0.453	0.152
Grammar Corrected	0.422	0.162

Model	WER	CER
Combined Corrected	0.375	0.141

The CER stayed consistently lower than the WER across all types of corrections. This makes sense because even if we change words during grammar correction, the number of character mistakes might not increase as much. The show that our model is making a real difference in understanding spoken language. But there's room to grow, especially in improving the grammar correction without increasing word errors. We could be working on fine-tuning the balance between spelling and grammar corrections to get even better at understanding and transcribing speech accurately in future works.

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

In our project, we took on the challenge to help people who are learning English understand and write the language better. We didn't just think about it; we actually built a tool that can listen to speech and then write it down, checking for spelling and grammar mistakes along the way. We started by collecting real talks from TED and YouTube to make sure our tool could handle the way people really speak.

At the core of our system are three pre-trained models we picked out from Hugging Face. We made these programs better using the Torch framework for fine-tuning phase. The `facebook/wav2vec2-base-960h` was great for writing down what people said. The `oliverguhr/spelling-correction-English-base` and `vennify/t5-base-grammar-correction` then helped fix any spelling and grammar errors. Through fine-tuning and hyperparameter adjustments, our final model could really understand the language we found in our data, and the evaluation both in WER and CER decreased significantly, which indicated that our model had already enhanced the output.

Our web application, with its intuitive interface and responsive design, extends the reach of our voice-to-text modeling, making it easily accessible to students and educators around the world. It is more than just a technical achievement; it is a step towards inclusivity in education and a tool for global connection.

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, which caused us recognized why many public models struggle with them or choose to ignore them. 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.

References

- facebook/wav2vec2-base-960h
- oliverguhr/spelling-correction-english-base
- vennify/t5-base-grammar-correction