Speech to Text Transformation for Communication Impaired People

Muhammet Batuhan Doğan
Hacettepe University
Department of Artificial Intelligence Engineering
Ankara, Turkey
doganmuhammetbatuhan@gmail.com

Alper Mert
Hacettepe University
Department of Artificial Intelligence Engineering
Ankara, Turkey
alperm76@gmail.com

ABSTRACT

Effective communication is a fundamental human right, yet individuals with speech disorders, neurological conditions, or hearing impairments often face significant challenges in daily interactions. Speech-to-Text (STT) technology offers a promising avenue to bridge this communication gap by converting spoken language into written text. This project focused on developing an accessible and user-friendly STT system tailored to enhance communication for these individuals. The approach involved a two-fold strategy: first, the exploration and evaluation of speech recognition models, including a custom-trained Wav2Vec2ForCTC model and the pre-trained OpenAI Whisper model; and second, the development of an intuitive web-based user interface using Streamlit, prioritizing ease of use and broad device compatibility. Experimental results demonstrated the superior performance of the Whisper model, achieving a Word Error Rate (WER) of 0.27 compared to approximately 0.60 for the custom model on a specific test sentence. The developed interface facilitated easy audio input via file upload and direct microphone recording, proving to be accessible and efficient. This work culminates in a functional STT platform that holds the potential to improve the quality of life and promote inclusivity for people with communication impairments.

1 INTRODUCTION

People with communication impairments, such as those with speech disorders, neurological conditions, or hearing impairments, often encounter significant communication problems in their daily lives. These challenges markedly impact their ability to engage in social interactions, participate in professional environments, and communicate effectively in healthcare settings. Speech-to-Text (STT) technology presents a potential solution by converting spoken language into written text, potentially making communication more accessible for these individuals. The importance of addressing this problem cannot be overstated, as communication is a fundamental human right, and the absence of adequate accessibility tools can lead to social and economic barriers for affected individuals. Consequently, developing an efficient STT system specifically tailored to the needs of people with speech impairments can substantially improve their quality of life, foster greater inclusivity, and contribute to broader technological advancements in accessibility. This project aimed to address these challenges by developing and evaluating an STT system that is not only accurate but also accessible and user-friendly. Our approach involved exploring different speech recognition models to identify the most effective one and designing an intuitive user interface to ensure ease of use for the target population. This report details the methodologies employed, the experimental results obtained from model evaluations and user interface testing, and the overall conclusions drawn from this work.

Conference'17, July 2017, Washington, DC, USA
2025.

2 RELATED WORK

Paper 1: Speech to Text Conversion and Summarization for Effective Understanding and Documentation [1]

Authors: Vinnarasu A., Deepa V. Jose
Journal: International Journal of Electrical and Computer Engineering (IJECE), Vol. 9, No. 5, October 2019 This paper presents a hybrid model that combines speech-to-text conversion and text summarization to support efficient understanding and documentation of lengthy speech data. The main motivation is to automate the creation of concise summaries from spoken content such as lectures, presentations, or meetings, which would otherwise be time-consuming to transcribe and condense manually. The authors use Google Speech API to convert speech into raw text. However, a major limitation of the API is its inability to insert punctuation such as periods (.) and question marks (?), which are essential for segmenting the text into meaningful units. To solve this, the authors propose a custom pre-processing mechanism: periods and question marks are programmatically inserted based on pause durations. This improves sentence segmentation, thereby enhancing the quality of downstream summarization. The summarization part of the pipeline uses Python's Natural Language Toolkit (NLTK) for tokenization, word frequency analysis, and sentence ranking. Words with frequencies outside a specified range are removed to reduce noise. Sentences are then ranked based on their cumulative word frequencies, and Python's `heapq.nlargest()` function is used to select the most informative ones for the summary. The authors compare the performance of their method against the popular Gensim library. Their method is found to be faster and more flexible, allowing summaries of varying lengths, while Gensim typically produces only one-line summaries. Experimental results show that pre-processed text (with punctuation) significantly improves both recognition time and summarization quality. The paper concludes that this model is practical for real-world applications, such as converting academic lectures into student-friendly notes.

Paper 2: Voice Recognition System: Speech-to-Text [2]

Authors: Prerana Das, Kakali Acharjee, Pranab Das, Vijay Prasad
Journal: Journal of Applied and Fundamental Sciences, Vol 1(2), November 2015 This paper focuses on a system that converts human speeches to text. The main problem addressed is the need for an accurate, real-time speech-to-text system that can understand user voice commands or dictations without relying on a physical keyboard. This is particularly relevant for accessibility, home automation, and hands-free computing applications. The proposed system follows a structured pipeline:

a) Acoustic Signal Processing: The system records the user's voice using a microphone.
b) Feature Extraction: It uses Mel Frequency Cepstral Coefficients (MFCC) to capture perceptually relevant characteristics of the speech signal.
c) Feature Matching: Implements Vector Quantization (VQ) to group and compare speech patterns.
d) Modeling and Classification: Applies Hidden Markov Models (HMMs) to model temporal dynamics of speech and recognize phonemes and words.
e) Decoding: Converts matched features into interpretable text.
f) Output: Displays the recognized text to the user.

The authors emphasize the modular design of the system, including training and testing phases. Training data is pre-processed and stored in .mat files using MATLAB, which also serves as the platform for interface and testing. The paper also reviews related research from 2004 to 2013, evaluating alternative feature extraction and filtering techniques such as Dynamic Time Warping (DTW), FIR filtering, and Kalman filters, highlighting their contributions to speech denoising and recognition robustness. The results show that combining MFCC with HMM and VQ leads to effective speech recognition with good speed and reasonable accuracy. The system shows promise in real-world applications, especially in developing automated systems like voice-controlled smart homes. The authors conclude that further research should explore larger vocabularies, continuous and spontaneous speech, and multilingual support.

Paper 3: Deaf Chat: A Speech-to-Text Communication Aid for Hearing Deficiency [4]

Authors: Mandlenkosi Shezi, Abejide Ade-Ibijola
Journal: Advances in Science, Technology and Engineering Systems Journal, Vol. 5, No. 5 (2020) This paper introduces Deaf Chat, an AI-powered speech-to-text communication tool designed to assist people with hearing impairments. The problem addressed is the limited effectiveness of existing speech recognition tools, which often fails to adapt for multiple speakers or are restricted to specific scenarios or types of hearing impairment (e.g., only for the deaf or hard-of-hearing, but not both). Deaf Chat seeks to overcome these limitations by incorporating Multiple Speaker Classification and real-time speech diarization capabilities. The tool uses the IBM Watson Speech-to-Text API, which includes speaker recognition and noise filtering using deep neural networks. The authors developed a mobile app using Android Studio and Java/XML that allows users to record voice notes, which are then converted into segmented, speaker-classified text. The result is an accessible chat application where hearing users can communicate with hearing-impaired individuals seamlessly. The algorithm behind Deaf Chat separates speech inputs by speaker before transcribing them. The user can then edit and send this text. The authors conducted a survey to evaluate usability, accessibility, and performance. Results showed: Over 83% of users would recommend it to hearing-impaired individuals. 86% found the speech-to-text accurate, and 88.5% found speaker separation effective. Some feedbacks show that the model is not fully efficient with non-English languages.

3 THE APPROACH

Our project methodology was systematically divided into several key stages to address the challenge of speech-to-text transformation for communication-impaired individuals. This involved creating and evaluating a custom speech recognition model, assessing a state-of-the-art pre-trained model, and subsequently developing an accessible user interface for the most effective model.

3.1 Dataset Preparation

Rather than using a predefined dataset, a custom dataset was curated for training and testing the speech recognition models. This dataset comprised:

- Five YouTube videos featuring speeches by Benedict Cumberbatch [7–11], chosen for his clear and fluent English articulation. The corresponding transcripts were also utilized.
- Our own voice recordings, which were primarily used to test the developed models.

3.2 Speech Recognition Model Exploration

3.2.1 Custom Trained Model: Wav2Vec2ForCTC. Our initial approach involved fine-tuning a contemporary speech recognition model.

- **Model Loading:** The Wav2Vec2ForCTC model was loaded from the HuggingFace model hub.
- **Data Preparation:** Training data was meticulously prepared, consisting of `.wav` audio files and their corresponding `.txt` transcription files. The five YouTube videos and their transcripts [7–11] formed the core of this training set. All audio files were resampled to 16kHz monophonic sound to ensure consistency.
- **Initialization:** A tokenizer and feature extractor were initialized using the `Wav2Vec2Processor`.
- **Fine-tuning:** The model was fine-tuned using the HuggingFace `Trainer` API, leveraging its efficient training loops and utilities.
- **Prediction Mechanism:** During the prediction phase, input audio values were fed to the model to compute logits. Predicted character IDs were then extracted from these logits using an `argmax` operation, and these IDs were subsequently converted into human-readable text via the processor's `decode` method.

3.2.2 Pre-trained Model: OpenAI's Whisper. Recognizing the advancements in large-scale pre-trained models, we also integrated and evaluated OpenAI's Whisper model.

- **Rationale:** This model was selected due to its renowned accuracy and extensive language capacity.
- **Robustness:** Whisper is known for its robust performance, even under challenging conditions such as noisy environments, accented speech, or otherwise imperfect audio inputs.

4 FRONTEND/USER INTERFACE (UI) DEVELOPMENT

To ensure the practical applicability and accessibility of our speech-to-text solution, significant effort was dedicated to developing a user-friendly frontend interface.

4.1 Framework Selection

We selected Streamlit [5] as the primary frontend framework. This choice was driven by its rapid prototyping capabilities, inherent cross-platform compatibility, and the ease with which it integrates with Python-based machine learning backends. Streamlit facilitates the seamless deployment of interactive data applications, negating the need for extensive traditional frontend development experience.

4.2 UI Layout and Functionality

The user interface was designed to be intuitive and was structured to accommodate two primary functionalities, as depicted in Figure 1:

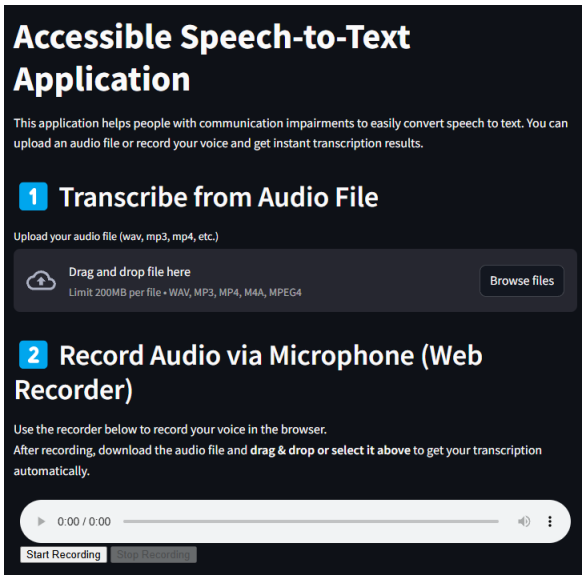

Figure 1: User interface of the Accessible Speech-to-Text Application, showing options for file upload and microphone recording.

- **Audio File Upload and Transcription:** Users can upload pre-recorded audio files in various formats (wav, mp3, mp4, m4a) directly through a drag-and-drop uploader interface component. Upon successful upload, the selected file is automatically processed by the backend speech-to-text model, and the resulting transcription is promptly displayed on the page.
- **Microphone-Based Recording and Transcription:** To cater to users who may not have pre-recorded audio or prefer live input, we embedded an HTML5 audio recorder [3] directly into the interface. This feature enables users to record their voice using their device's microphone within the browser environment. They can then download this recording and subsequently upload it for immediate transcription through the system.

4.3 Design Principles

Our UI design was fundamentally governed by the principles of accessibility and simplicity, drawing from established guidelines [6]:

- **Clear Instructions:** Each functional section of the interface is accompanied by concise and easy-to-understand user instructions, ensuring clarity on how to interact with the system at every step.
- **Minimalist Layout:** The interface is intentionally kept clean and uncluttered, minimizing potential distractions by focusing squarely on core functionalities and eliminating superfluous controls or visual decorations.
- **Feedback and Status Updates:** The system is designed to provide users with instant feedback through visual cues such as loading spinners, status messages for ongoing processes, and clear error reporting, which helps users understand the current state of their request and its outcome.

4.4 Technical Implementation Details

- The audio upload and transcription display features leverage Streamlit's native `st.file_uploader` and `st.text_area` components [5].
- **Automatic Transcription Trigger:** The system is configured to automatically initiate the transcription process as soon as a file is uploaded, thereby eliminating extra user actions and streamlining the overall user experience.
- The HTML5 audio recorder is integrated into the Streamlit application using the `st.components.v1.html` function, which allows embedding custom HTML, JavaScript, and CSS. This leverages browser APIs for in-place audio recording and playback without imposing additional software dependencies on the user [3].
- **Privacy and Security:** For security and to protect user privacy, all uploaded and recorded audio files are processed temporarily by the system and are deleted from the server after the transcription process is completed.

4.5 Accessibility and Inclusivity

The UI has been designed with the explicit goal of being operable by users who may have limited technical proficiency and/or physical limitations, following accessibility best practices [6]. No advanced configuration or software installation is required on the client-side; all core functionalities are browser-based and only necessitate microphone permission from the user for the recording feature.

Finally, the insights gained from evaluating both the custom-trained and pre-trained models informed the decision for which

model to integrate with this GUI, aiming for the best overall performance and user experience.

5 EXPERIMENTAL RESULTS

This section presents a comprehensive analysis of the performance of the speech recognition models developed and evaluated, alongside an assessment of the user interface's effectiveness and usability.

5.1 Performance of the Custom Trained Model (Wav2Vec2ForCTC)

The Wav2Vec2ForCTC model, fine-tuned on our custom dataset, was tested using a specific English voice recording.

- **Test Case:**
  - *Ground Truth:* "HI I AM ALPER I AM 23 YEARS OLD I AM AN AI STUDENT AT HACETTEPE UNIVERSITY THIS IS OUR SPEECH TO TEXT PROJECT FOR MACHINE LEARNING FOR HEALTHCARE COURSE LETS SEE HOW ACCURATE THE MODEL IS"
  - *Predicted Text:* "HA I AM MY PAR I AM TWENTY THRE YEARS OLD I AM ANA A SUTONTE TA YOR TEPEUNIVERSITY THIS IS OUR SPECI TO TAXE PROJECT FOR MISHION EARNING FOR AT GAR COURSE LET'S SE HOW I GRET THE MOTHER IS"
- **Word Error Rate (WER):** The model achieved a Word Error Rate (WER) of approximately 0.595, or 59.5%. This indicates a significant mismatch between the predicted and actual text.
- **Evaluation:**
  - While the model demonstrated an ability to capture some core phonetic patterns, many words were misrecognized or confused with similar-sounding terms.
  - The limited scope of the training data, primarily sourced from a single speaker, likely hindered the model's capacity to generalize to other voices or nuances effectively.
  - Notable distortions in word endings (e.g., "student" transcribed as "sutonTe") suggested potential instability at the character-level recognition.

5.2 Performance of the Pre-trained Model (OpenAI's Whisper)

OpenAI's Whisper model was evaluated using a variety of data to assess its capabilities.

- **Testing Data:** The model was tested with the same YouTube videos [7–11] used for training the Wav2Vec2ForCTC model, as well as our own voice recordings in three different languages: Turkish, English, and German.
- **General Performance:** Across all tested videos, the Whisper model performed well. It demonstrated robust transcription capabilities for our voice recordings in Turkish, English, and German, with the predicted text closely mirroring the original speech. Most discrepancies were minor, relating to punctuation or slight letter variations.
- **Comparative Performance:** On the same English test sentence used for the Wav2Vec2ForCTC model, the Whisper model achieved a significantly lower Word Error Rate (WER) of 0.27. This was a substantial improvement over the custom model's 0.60 WER (approximately, from 0.595) on the same audio.

5.3 User Interface (UI) Evaluation

The developed frontend interface underwent usability testing, focusing on its core functionalities and accessibility.

- The file upload and automatic transcription pipeline was observed to minimize user effort and maximize accessibility.
- Embedding an HTML5-based audio recorder [3] directly into the interface successfully resolved browser compatibility issues that are frequently observed with Python-side audio recording solutions, thereby offering a more robust and stable user experience.
- The interface was tested on both desktop and mobile browsers, confirming its usability and accessibility across different platforms.

5.4 Conclusion from Experimental Evaluation

Based on the starkly superior performance of the Whisper model in terms of accuracy (WER of 0.27 vs. 0.60) and its robust handling of multilingual input, it was a clear decision to select the Whisper model for integration into the final GUI application. The user interface evaluations also confirmed that the chosen design and technologies provided an accessible and efficient platform for users.

6 LIMITATIONS

While the project successfully developed a functional speech-to-text system with an accessible user interface, several limitations were identified during development and evaluation. These pertain to both the speech recognition models explored and the implemented frontend.

6.1 Model Limitations

- **Custom Trained Model (Wav2Vec2ForCTC):**
  - The performance of the custom-trained Wav2Vec2ForCTC model was significantly hampered by the limited scope and diversity of the training data, which was primarily sourced from a single speaker. This likely led to its failure to generalize effectively to other voices or speaking styles.
  - The model exhibited instability at the character-level recognition, as evidenced by distortions in word endings (e.g., "student" being transcribed as "sutonTe").
  - Consequently, the model yielded a high Word Error Rate (WER) of approximately 59.5%, indicating a substantial number of inaccuracies in its transcriptions.
- **Pre-trained Model (OpenAI's Whisper):**
  - Although OpenAI's Whisper model demonstrated significantly better performance, our implementation utilized the general-purpose, out-of-the-box version. While effective, it was not specifically fine-tuned on a dataset tailored to the unique speech patterns of individuals with diverse communication impairments. The suggestion to potentially apply further data augmentation, fine-tuning, and segmentation implies that its current state, while good, could be further optimized for specific needs.

6.2 User Interface Limitations

- **Workflow for Microphone Recording:**
  - The microphone-based recording feature, while functional, does not offer full automation in the sense of direct transcription immediately after a recording is stopped within the browser. Users are currently required to manually download the recording made via the HTML5 recorder and then upload this file to the system for transcription. This multi-step process is a consequence of browser security limitations and Streamlit's current architectural constraints regarding direct audio stream handling from the browser to the Python backend in this specific embedded component setup.

These identified limitations provide clear directions for potential enhancements and future work, which will be discussed in the Conclusion section.

7 CONCLUSIONS

This project successfully addressed the challenge of developing an accessible Speech-to-Text (STT) system for individuals with communication impairments by evaluating speech recognition models and creating a user-friendly interface. The investigation confirmed OpenAI's Whisper model as significantly more accurate (0.27 WER) than a custom-trained Wav2Vec2ForCTC model (approx. 0.60 WER), leading to its adoption in the final Streamlit-based application. The resulting platform offers an effective and accessible means of converting speech to text, demonstrating the potential to enhance communication for the target user group.

Key directions for future work include:

- **Model Optimization:** Further fine-tuning the Whisper model on datasets specific to diverse communication impairments to enhance its specialized accuracy.
- **User Experience Refinement:** Implementing full automation for the microphone-to-transcription workflow to improve ease of use, possibly through custom components or advanced web technologies, and conducting extensive usability testing with the target community.
- **Functionality Expansion:** Exploring valuable additions such as text summarization to increase the system's utility.

In essence, this work provides a robust foundation for an assistive STT tool, with clear pathways for future enhancements that promise to further empower individuals facing communication challenges.

REFERENCES

[1] Vinnarasu A. and Deepa V. Jose. 2019. Speech to Text Conversion and Summarization for Effective Understanding and Documentation. *International Journal of Electrical and Computer Engineering (IJECE)* 9, 5 (October 2019). https://www.researchgate.net/publication/342147736

[2] Prerana Das, Kakali Acharjee, Pranab Das, and Vijay Prasad. 2015. Voice Recognition System: Speech-to-Text. *Journal of Applied and Fundamental Sciences* 1, 2 (November 2015). https://www.researchgate.net/publication/304651244

[3] Mozilla Developer Network. 2025. *MediaRecorder API.* Retrieved May 17, 2025 from https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder_API

[4] Mandlenkosi Shezi and Abejide Ade-Ibijola. 2020. Deaf Chat: A Speech-to-Text Communication Aid for Hearing Deficiency. *Advances in Science, Technology and Engineering Systems Journal* 5, 5 (2020).

[5] Streamlit Team. 2025. *Streamlit Documentation.* Retrieved May 17, 2025 from https://docs.streamlit.io/

[6] World Wide Web Consortium (W3C). 2025. *Web Accessibility Guidelines (WCAG) Overview.* Retrieved May 17, 2025 from https://www.w3.org/WAI/standards-guidelines/

[7] YouTube. [n. d.]. Benedict Cumberbatch Speech Short.

[8] YouTube. 2025. Benedict Cumberbatch Speech Video 1. https://www.youtube.com/watch?v=9rwN1W4WVI0. Access: May 17, 2025.

[9] YouTube. 2025. Benedict Cumberbatch Speech Video 2. https://www.youtube.com/watch?v=-VtzR9l3QJw. Access: May 17, 2025.

[10] YouTube. 2025. Benedict Cumberbatch Speech Video 3. https://www.youtube.com/watch?v=U1NtMRguvno. Access: May 17, 2025.

[11] YouTube. 2025. Benedict Cumberbatch Speech Video 4. https://www.youtube.com/watch?v=xVVwNzx7elE. Access: May 17, 2025.