

Using Deep Learning and Augmented Reality to Improve Accessibility: Inclusive Conversations Using Diarization, Captions, and Visualization

Mr. Yun Wang, Undergraduate at University of Illinois Urbana-Champaign

Yun Wang is an Undergraduate at University of Illinois Urbana-Champaign. Research interests include using technology, algorithms to improve accessibility and inclusive education.”

Mr. Colin P. Lualdi, University of Illinois Urbana-Champaign

Dr. Lawrence Angrave, University of Illinois Urbana-Champaign

Dr. Lawrence Angrave is an award-winning computer science Teaching Professor at the University of Illinois Urbana-Champaign. He creates and researches new opportunities for accessible and inclusive equitable education.

Guru Nanma Purushotam

Using Deep Learning and Augmented Reality to Improve Accessibility: Inclusive Conversations using Diarization, Captions, and Visualization

Abstract

The problem of diarization - identifying different speakers in a conversation stream - has not been sufficiently addressed for deaf and hard-of-hearing students in learning communities such as student design teams in engineering and related STEM disciplines. Though the accuracy of the latest automated real-time speech-to-text systems is now approaching usable low word error rates, the generated text output is an incomplete representation of a multi-party conversation; In short, it solves the “what” but not the “who.” This creates barriers to our ideal of an inclusive and equitable learning community. Thus students who are deaf or hard of hearing are further marginalized and excluded from multi-party peer discussions with non-deaf participants because it is hard to visually follow who is speaking. To address these communication barriers, we utilized the Human Centered Engineering Design framework to identify a set of features that overcomes the above barriers. This paper explores computerized diarization techniques that utilize a wide set of algorithms and audio metrics to assist in speaker identification. These techniques include mel-frequency cepstrum coefficients (MFCC), volume, fundamental frequency identification, and deep learning of voice prints. For the goals described in this paper, a subset of existing algorithms that respected privacy and legal constraints was selected and evaluated for the purposes of identifying speakers using a live audio stream. Several visualization methods were also designed and evaluated. These included visualization of embedding mel-frequency cepstrum, speaker identifier, pitch, volume, and other voice characteristics into a live caption stream. Both diarization and visualization were integrated into a live captioning tool, ScribeAR, previously introduced in ASEE regional proceedings, and rendered using a lightweight Augmented Reality display. In order to facilitate captioning services in areas with limited network connectivity, whisper.cpp, a derivative of OpenAI’s Whisper project, was also incorporated into the application. Links to the open source project are included so that other educators may adopt this inclusive practice. Some accessibility-related opportunities that could be used as motivating design projects for engineering students are described.

1. Introduction:

Live-captioning with augmented reality (AR) headsets is an effective and promising communication tool for students who are deaf or hard of hearing (DHH) [1]. Compared to basic live-captioning on a separate display, which causes information gaps for DHH students [2] resulting in lower academic performance [3], the use of AR headsets offers a more immersive and integrated experience for DHH students. However, the benefits of AR captioning might not

be fully realized without certain features, preventing DHH students from fully participating in structured and ad hoc class discussions and potentially negatively impacting their learning and engagement.

To address this challenge and enhance DHH students' educational experience, new features were explored for inclusion into ScribeAR [4], a free open-source web-based captioning tool, to include speaker diarization and contextual visualization. By using the techniques of Human Centered Engineering Design (HCED) [5] we incorporated three new features, which are discussed in this paper:

- Speaker diarization - to identify individual speakers in real time, enabling DHH students to better understand and participate in class discussions and lectures.
- Contextual visualization - these techniques can provide visual representations of the speech data, including emotional content, voice activity detection, and frequency.
- whisper.cpp [17] - a speech-to-text model that excels at generating natural-sounding text and has several benefits such as improved conversational flow, contextual understanding, and robustness against noise.

Our vision is to create a fully accessible, open-source solution that can provide a more inclusive and engaging learning experience for DHH students in engineering education settings. Furthermore, we aim to inspire and assist educators and students interested in starting similar accessibility projects. We include a step-by-step guide to help others interested in using or developing accessible technology within the context of engineering education. In the latter case, development of accessible technology or contribution of new accessible features could be configured as a compelling student project for independent study, capstone or similar course, or used as motivating context for undergraduate research. The source for this project is free and available under an open source license on our GitHub repository, [scribear/ScribeAR.github.io](https://github.com/scribear/ScribeAR).

Though it is possible to run machine models on remote servers, our experience with ScribeAR has shown that for accessible AR tools there is value in creating accessible technology that can function locally (e.g., for scenarios where with limited Internet connectivity) or at zero cost [4]. An additional advantage of local computation is that it scales; the user does not need to wait for captions or other results because a remote server is busy or overloaded or unavailable or requires new access keys. Previous work on ScribeAR enabled users to choose from several best-in-class remote speech-to-text engines; this feature can be extended to include local options. We developed ScribeAR following the HCED framework, which has five taxonomy spaces: Understand, Synthesize, Ideate, Prototype, and Implement. ScribeAR was created through multiple activities including interviews, brainstorming, prototyping, and testing in the spaces of the HCED framework, aiming to meet the needs of our users (DHH students). After implementing the first version of our tool, we evaluated it with user feedback and revised it accordingly. We repeated this process until we achieved a satisfactory product.

In this paper, we continue to explore the boundaries of web-based computation and assume the user may have only minimal (e.g., smartphone) or moderate (e.g., laptop) hardware available. The following sections discuss each feature in depth and finally the preliminary survey results are reported.

2. Speaker Diarization:

Speaker diarization is a technique that splits an audio stream into segments by speaker identity. It can enhance accessibility and inclusivity for DHH students in educational settings with multiple speakers, such as group discussions or panel presentations. It can also reduce social and communication barriers for DHH students in their interactions with classmates and course staff.

Our speaker diarization system consists of two components:

1. An embedding component that uses the generalized end-to-end (GE2E) loss [6] to segment the speech data and create a compact representation for each segment.
2. A clustering component called Links, introduced in [7]. This algorithm assigns a speaker label to each speech segment based on its similarity to previous utterances. The label can be either a new one or an existing one. The labels can be either generic, such as “Speaker 1” and “Speaker 2”, or specific, such as “Jo” and “Ahmed”.

2.1. Speaker Embedding - GE2E Loss

Speaker embedding is a way of capturing speaker characteristics from speech signals and encode them as low-dimensional vectors. Two main approaches for speaker embedding are i-vector (identity vector) [6] and d-vector (deep-neural-net vector or discriminative vector) [7]. I-vector uses factor analysis to reduce the dimensionality of speaker features, while d-vector uses deep neural networks to learn discriminative embeddings. D-vector has outperformed i-vector in accuracy and robustness.

GE2E loss is a loss function for training d-vector models. It makes the embeddings of the same speaker closer and those of different speakers farther apart. It leads to more precise speaker embedding models [8], surpassing those trained with the Tuple-Based End-to-End (TE2E) loss [9].

For our speaker diarization system, We chose the “d-vector V1” model from [8]. It is a pretrained Pytorch model [10] [11] in python, with a size of 37 MB, using GPU PyTorch library. The model can be quantized to less than 20 MB, which is suitable for low-resource settings and meets our project requirements.

We converted the PyTorch model into browser-based models and a repeatable methodology for this process is described in the appendix of this paper.

2.2. Online Clustering - The Links Algorithm

Online clustering, specifically for speaker diarization, is the process of assigning speaker identities to audio segments data that arrive in a stream. The state-of-the-art online algorithm for this task is UIS-RNN [8], which beats even the best offline algorithms. However, UIS-RNN is computationally costly. We adopted a simpler and faster online algorithm, Links [12], which groups unit vectors in high-dimensional Euclidean space without future embeddings or backtracking. Links can efficiently cluster streaming data on the fly.

The Links algorithm maintains a two-level hierarchy of clusters and subclusters. When a new speech segment is available, the algorithm first assigns it to the most similar subcluster by calculating the cosine similarity between the segment's embedding vector (generated using the GE2E model) and the subcluster's centroid. The subcluster with the highest similarity score is selected as the most similar subcluster for the input vector.

Then, using the selected subcluster's centroid, the algorithm calculates the most similar cluster by computing the cosine similarity between the subcluster's centroid and the centroids of all existing clusters. The cluster with the highest similarity score is selected as the most similar cluster for the input vector.

However, to achieve optimal performance, the similarity thresholds (T_c , T_s , and T_p) in the Links algorithm need to be tuned based on the specific speech audio data source. This can be done by manually labeling a speech audio dataset with cluster IDs, running the cluster algorithm on the data, and adjusting the similarity thresholds to achieve high accuracy. The accuracy is evaluated as the ratio between correct and incorrect cluster IDs. In our case, we used the recording of two speakers (a low pitch male voice and a high pitch female voice) to tune the similarity thresholds.

3. Contextual Visualization:

Contextual visualization is an essential component of ScribeAR, which aims to provide DHH students with informative visuals that enable them to better understand their surroundings. A previous iteration of the tool described in [4] offered waveform (time-domain) and circular (frequency-domain) audio visualizations. Following user feedback, we made the following enhancements to the visualizations (Fig 1.):

1. Implemented a high color contrast to improve visibility.
2. Allowed the visualizations to be draggable and resizable.
3. Added frequency labels (in kilohertz) to the circular frequency-domain visualization.

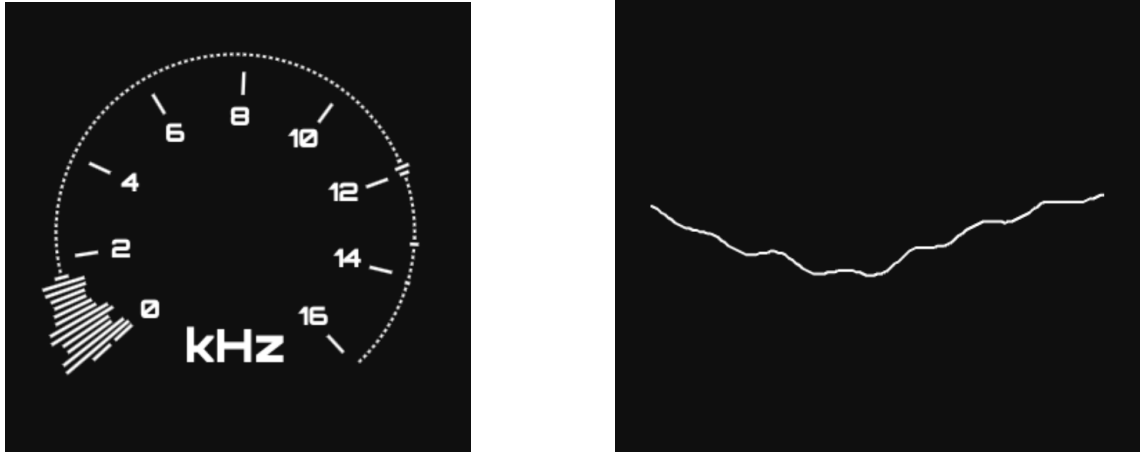
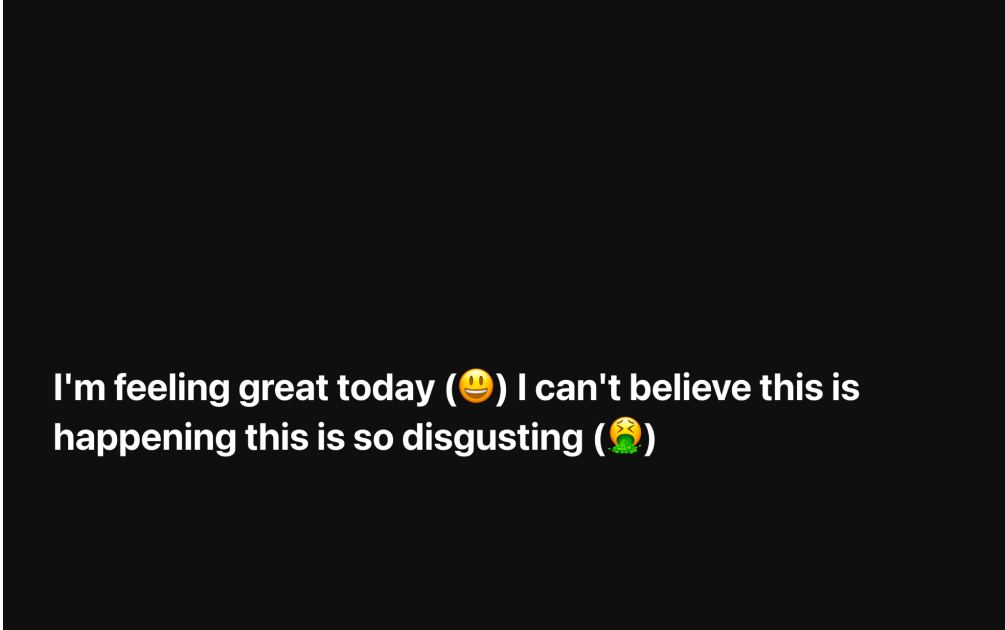


Figure 1. Screen captures of the upgraded ScribeAR audio visualizations. Two types are offered. *Left*: Circular visualization displaying frequency-domain data. *Right*: Waveform visualization displaying time-domain data.

Next, we will discuss two new types of contextual visualization we added to ScribeAR. The first is an emoji visualization to indicate the emotional context of the transcribed speech as determined by sentiment classification models. The second is a MFCC visualization.

3.1 Intent/Emotions Visualization

An interesting design problem is how to visualize and present the emotional cues. In order to minimize distractions and avoid introducing an additional visual element to monitor, this paper presents a novel approach that embeds emojis directly into the caption display area. Furthermore, the emoticons serve as a summarized representation for users who desire a rapid, visual sentiment summary. This summary can be useful when the user is busy and thus not closely processing written captions, or looking to resume or re-integrate into the conversation. The selection of the appropriate emoji is performed by an intent recognition machine learning model to visualize the emotional intent (e.g., joy, anger, frustration, neutral) of the spoken sentence (Fig 2.).



I'm feeling great today 😊 I can't believe this is happening this is so disgusting 🤢

Figure 2. Screen capture of our browser-based intent model and presentation functionality. The intent of each sentence is evaluated, then displayed using an emoji icon after the end of each sentence. The language model is able to extract intent at a sufficiently fast rate for real-time transcription.

Google’s BERT [13] language model was used in the browser to extract the emotional context (“intent”) from each sentence of the transcript. The ONNX Runtime [14], a cross-platform library that supported models from various frameworks, such as PyTorch, Tensorflow/Keras, TFLite, scikit-learn, etc. (source: ONNX website) was able to run BERT in a web browser. The `onnxruntime` tutorial [9] was used to adapt an intent classification deep learning tool to a web application. This was a relatively straightforward process.

With our experimental prototype, we were able to show the feasibility of this approach in preparation for the next evaluation step. However, at the time of writing, the model occasionally failed to load the machine learning models when integrating the prototype with ScribeAR,. We will continue to improve the integration with ScribeAR and will be publishing fixes and improvements to our publicly available source code on GitHub.

3.2 MFCC Visualization

Conventional, (non-AI) audio visualizations still provide valuable information to the user. For example, we explored the Meyda library [15] to offer log-scaled pitch and frequency visualizations in a web browser with fast refresh rates ($> 10\text{Hz}$) (Fig. 2).

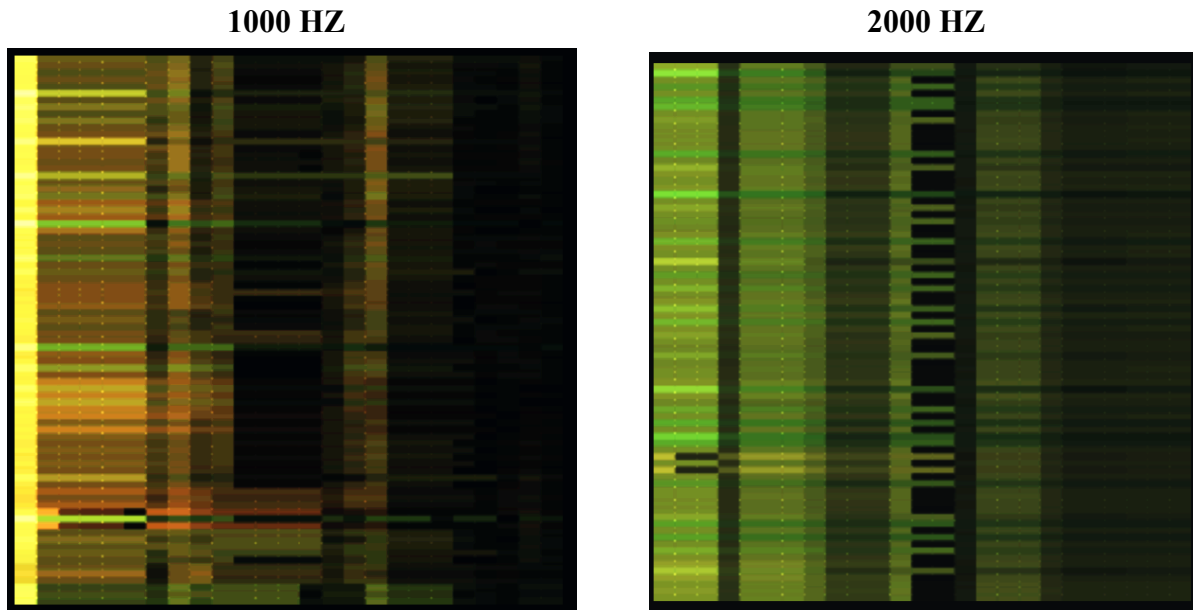


Figure 2. A lower pitch will appear red and orange, while a higher pitch will appear yellow and green, as HSL hue corresponds to the weighted average log pitch. Each row corresponds to a single moment, with a band of intensity corresponding to the mel-frequency cepstral coefficients - a classic conventional audio analysis quantity.

Though MFCC visualization was implemented as a precursor for deep-net contextual visualization, it also provides a view into the pitch and spectrum of sound that was otherwise lacking in caption-based tools.

4. Integration of whisper.cpp

The integration of OpenAI's Whisper Automatic Speech Recognition (ASR) system into our application has been made possible and cost-effective by the open sourcing of the Whisper deep learning model [16]. We have incorporated Whisper.cpp [17] into our web-based application designed primarily for educational purposes in STEM-related courses for the DHH community. By utilizing WebAssembly, common hardware capabilities available on a wide range of platforms can be leveraged, ensuring that the model runs efficiently on users' browsers.

WebAssembly's high-performance design aims to execute code at native speed, with its stack machine encoded in a size- and load-time-efficient binary format. This ensures that the Whisper ASR system can deliver live transcriptions even under low network connectivity, enabling DHH individuals to utilize our tool for educational activities regardless of their location. The Whisper model's ability to capture conversational flow and enhanced contextual understanding contribute

to the accuracy and inclusivity of our application, further facilitating communication access for the DHH community.

Despite its many advantages, integrating the Whisper model using WebAssembly presented some challenges. For instance, we had to enable SharedArrayBuffer memory through the developer's tool separately, as WebAssembly uses SharedArrayBuffer to share memory between WebAssembly threads. The implementation details to integrate whisper.cpp into a web-based application have been detailed in the appendix. By overcoming the technical challenges, an accurate and cost-effective captioning solution can be successfully incorporated, providing DHH individuals with a reliable and efficient tool for their educational needs in STEM-related courses.

5. Preliminary Survey

Preliminary feedback was gathered from student users on the inclusion of emojis (Fig. 2 and Fig. 3) and their impact on transcript comprehension. Their responses are below:

- “It's really cool to be able to understand jokes from different cultures without feeling left out of any conversation. This helps us feel more connected to each other.”
- “It's easy to understand the context and changes in tones, and to steer the conversation in a specific direction using certain words. I really like this feature.”

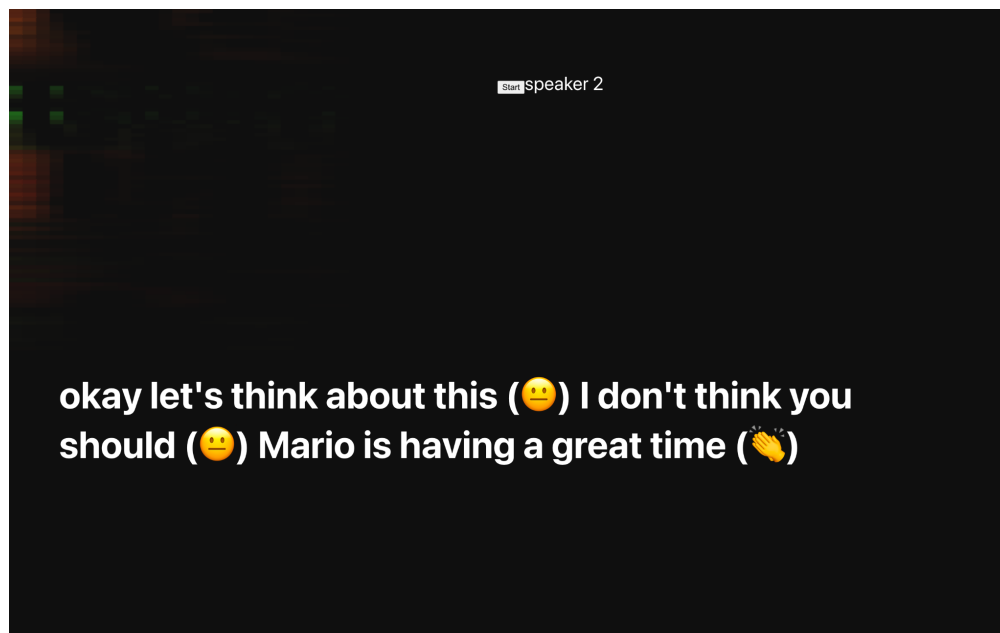


Figure 3. Screen capture of ScribeAR. *Top left*: The MFCC visualization. *Top center*: The current speaker as determined by diarization. *Center*: The transcribed speech with intent emojis.

6. Conclusion & Future Work

ScribeAR is a web-based live-captioning tool that enhances accessibility for Deaf and Hard of Hearing (DHH) engineering students by providing efficient speaker diarization and contextual visualization. In this paper, we present the design and implementation of our diarization system, which leverages the GE2E loss embedding model and the Links online clustering algorithm to achieve satisfactory accuracy and low latency. We also describe how ScribeAR offers contextual visualization of the ambient sound and the speaker's intent/emotion to enrich the information available to DHH students.

Our preliminary results show that our diarization system can perform basic voice diarization based on pitch. A future approach would implement and evaluate [8], which reported that the state-of-the-art embedding model “d-vector V3” was trained on data from far-field devices, public datasets including LibriSpeech [18], VoxCeleb [19], VoxCeleb2 [20], and non-public data. Custom speaker embedding models using the MultiReader training process would also be possible [7].

The current contextual visualization provides simple audio visualization and a basic intent/emotion visualization powered by a BERT tokenizer [13]. Other tokenizers or incorporate multi-modal data for intent/emotion visualization are possible. Adding whisper.cpp enhanced ScribeAR's accessibility. An Azure cloud solution has already been integrated into ScribeAR but other integration projects are possible. For example, ESPnet [21] is also a common speech to text platform that is an end-to-end speech processing toolkit. It includes various applications such as speech recognition, text-to-speech, speech translation, and speech enhancement.

Recent advances in Large Language Models will also provide new opportunities for inclusive conversational approaches. For example, a student project might use the new ChatGPT API that as of May 2023 is now available as part of Microsoft Azure cloud services, to provide summarization or other textual transformation of a transcript [24].

8. Acknowledgments

We thank the VR@Illinois program and the Department of Physics Graduate Office at the University of Illinois Urbana-Champaign for providing seed funding for this project. Portions of this research were also supported by a Microsoft 2022 award for accessible technology research and Azure cloud credits. Lastly, we thank current and former ScribeAR team members for their contributions to this project.

9. Appendix

9.1 Steps to integrate a PyTorch model into a web application

1. Obtain the speaker embedding PyTorch model trained using GE2E loss

```
model = torch.hub.load('RF5/simple-speaker-embedding',
    'gru_embedder')
```
2. Convert to ONNX

```
torch.onnx.export(model, input, "filepath.onnx")
```
3. Transform the input stream into appropriate input features for the model. For audio streams this pipeline typically requires three steps
 - a. A fast fourier transform (FFT) function and a window function step to convert audio stream into the frequency domain.
 - b. Mel filterbank: Compute mel filterbank given number of expected features, number of fast fourier transform size, sampling rate, minimum mel-log frequency, and maximum mel-log frequency.
 - c. Prepare a mel-log spectrogram: Given an input spectrogram and filterbank, use matrix multiplication to create a mel-log spectrogram with desired matrix shape.
4. Integrate into web application

Use `onnxruntime-web` which is a javascript machine learning package that runs machine learning models in the browser. For other models, one can also use `tensorflow.js`.

9.2 Steps to include whisper.cpp model into a web application

1. Enable `SharedArrayBuffer` memory by requesting an access token from chrome's developer site [22] to make the website cross-origin isolated. This is required to run WebAssembly threads which share memory between them

Inserted an `origin-trial` `<meta>` tag with the access token into the head section of each relevant webpage.
2. Use Emscripten [23] to convert C++ implementation of the whisper model [17] to wasm runtimes. These `stream.js` files utilized the WebAssembly module to run live transcription.

Emscripten command: `emcmake cmake && make`
3. Use an `IFRAME` element to read the live transcription from native javascript files, then display in the ScribeAR.

References

- [1] A. Miller, J. Malasig, B. Castro, V. L. Hanson, H. Nicolau, and A. Brandão, “The Use of Smart Glasses for Lecture Comprehension by Deaf and Hard of Hearing Students,” in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, in CHI EA '17. New York, NY, USA: Association for Computing Machinery, May 2017, pp. 1909–1915. doi: 10.1145/3027063.3053117.
- [2] R. Kushalnagar and P. Kushalnagar, “Collaborative gaze cues and replay for deaf and hard of hearing students,” in *Computers Helping People with Special Needs: 14th International Conference, ICCHP 2014, Paris, France, July 9-11, 2014, Proceedings, Part II 14*, Springer, 2014, pp. 415–422.
- [3] M. Marschark *et al.*, “Benefits of Sign Language Interpreting and Text Alternatives for Deaf Students’ Classroom Learning,” *J. Deaf Stud. Deaf Educ.*, vol. 11, no. 4, pp. 421–437, Oct. 2006, doi: 10.1093/deafed/enl013.
- [4] Angrave, L., Lualdi, C., Jawad, M., & Javid, T. “ScribeAR: A New Take on Augmented-Reality Captioning for Inclusive Education Access”. *2021 Illinois-Indiana Regional Conference*, 2021. <https://peer.asee.org/38276>
- [5] L. Lawrence, S. Shehab, M. Tissenbau, T. Rui, and T. Hixon, “*Human-Centered Design Taxonomy: Case Study Application With Novice, Multidisciplinary Designers*”. 2020. doi: 10.3102/1690347.
- [6] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-End Factor Analysis for Speaker Verification,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, no. 4, pp. 788–798, May 2011, doi: 10.1109/TASL.2010.2064307.
- [7] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, “Generalized End-to-End Loss for Speaker Verification,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, pp. 4879–4883. doi: 10.1109/ICASSP.2018.8462665.
- [8] C. Wang, A. Zhang, Q. Wang, and Z. ZHU, “Fully supervised speaker diarization,” US11031017B2, Jun. 08, 2021 Accessed: Apr. 09, 2023. [Online]. Available: <https://patents.google.com/patent/US11031017B2/en>
- [9] J. K. Bergum, “Text Emotion Prediction in Browser.” <https://github.com/jobergum/browser-ml-inference>
- [10] M. Baas, “Simple speaker embeddings.” <https://github.com/RF5/simple-speaker-embedding>
- [11] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2019. Accessed: Apr. 09, 2023. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html
- [12] P. A. Mansfield, Q. Wang, C. Downey, L. Wan, and I. L. Moreno, “Links: A High-Dimensional Online Clustering Method.” arXiv, Jan. 30, 2018. doi: 10.48550/arXiv.1801.10123.
- [13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.
- [14] “ONNX RUNTIME.” <https://onnxruntime.ai/>
- [15] “meyda.” <https://github.com/meyda/meyda>
- [16] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust Speech Recognition via Large-Scale Weak Supervision.” arXiv, Dec. 06, 2022. doi:

- 10.48550/arXiv.2212.04356.
- [17] G. Gerganov, “whisper.cpp,” *GitHub repository*, 2023.
<https://github.com/ggerganov/whisper.cpp>
 - [18] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR corpus based on public domain audio books,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 5206–5210. doi: 10.1109/ICASSP.2015.7178964.
 - [19] A. Nagrani, J. S. Chung, and A. Zisserman, “VoxCeleb: a large-scale speaker identification dataset,” in *Interspeech 2017*, Aug. 2017, pp. 2616–2620. doi: 10.21437/Interspeech.2017-950.
 - [20] J. S. Chung, A. Nagrani, and A. Zisserman, “VoxCeleb2: Deep Speaker Recognition,” in *Interspeech 2018*, Sep. 2018, pp. 1086–1090. doi: 10.21437/Interspeech.2018-1929.
 - [21] S. Watanabe *et al.*, “ESPnet: End-to-End Speech Processing Toolkit.” arXiv, Mar. 30, 2018. doi: 10.48550/arXiv.1804.00015.
 - [22] “SharedArrayBuffer updates in Android Chrome 88 and Desktop Chrome 92.”
<https://developer.chrome.com/blog/enabling-shared-array-buffer/#origin-trial>
 - [23] “Emscripten.” <https://emscripten.org/>
 - [24] “ChatGPT”
<https://learn.microsoft.com/en-us/azure/cognitive-services/openai/how-to/chatgpt>