Automatic Speech Recognition for Second Language Teaching

Harsha Tummala, Andrew Liu, Yitong Chen and Mohamed Abdelmoneum
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In our work, we discuss approaches to improve automatic speech recognition (ASR) performance for non-native speakers learning English. Our goal is to build ASR systems that work equally well across native and non-native accents. We review prior work, including fine-tuning on accented data, transfer learning, and multi-task learning. We propose benchmarking ASR performance across accents using the Speech Accent Archive and AccentDB datasets. We will start with an off-the-shelf Whisper model and also explore accent classification and neutralization with the AccentDB model. We focus specifically on using ASR to support second language teaching. We emphasize optimizing models to analyze contributions of accent to recognize errors, and give feedback to learners. Our approach classifies accents with a convolution neural network, and then fine-tunes the Whisper model separately on multiple accents from AccentDB. Next steps are rebuilding raw audio from the AccentDB feature set and extending to other accented speech datasets. Overall we aim to customize ASR to individual accents in order to improve performance, support communication across accents, and provide tools for second language pedagogy.

1 Introduction

Automatic speech recognition (ASR) systems are increasingly a gateway to accessing information and services. Many customer service resources, for example, require users to go through an automated speech interface before having the opportunity to connect with a person. Behind the scenes systems like the 3M M*Modal virtual assistant support and guide doctors in their interactions with patients. We need ASR to be working well and equitably. There are disparities, however, in how well ASR systems perform for speakers with "non-standard" speech characteristics such as non-native accent. This is certainly a concern from an inclusion point of view. As machine learning systems may be trained largely using data collected from speakers who have had successful interactions with the systems, it is also a statistical concern. The effect of accent on ASR performance is complicated to define, but Google, for example, reports a word error rate (WER) of 24% on its cloud system [15]. In this project, we aim to improve English ASR performance for non-native speakers. For a selection of systems, we will attempt to replicate reported results and then build on them, hopefully finding ways to improve performance for this target group. We will benchmark results for multiple training sets. We will then focus on fine-tuning approaches, potentially using both single-accent and multiple-accent training data.

2 Literature Review

We must first define what we mean by accent. One speaker's pronunciation can vary from another's in many ways and for many reasons. We all have an accent from someone's point of view[10]. For this study, we are considering speakers whose English is influenced by a different language, usually their

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Site	Google[15]	Amazon[5]]	Facebook[6]	IIT	UBC[14]
Model	RNN-T	HMM-LSTM		TDNN	Wav2vec
Approach	Fine-tuning	Transfer learning / phoneme map	GAN (PT) + LAS (FT)	Multi-task	Transfer learning
Training Data set	L2-Arctic	Unified 4-dialect with phone mapping	Crowdsourced, 9-accent	Combined 7-dialect Commonvoice	Librispeech
Training data hours	18	40000	3800	34	960
Test data set	L2-Arctic	4-dialect, divided	1.2K utterances per accent	Indian English Commonvoice	960
Test data hours	2	2000	10800	1.3	1
Base WER	24 (cloud) 13.3 (RNN-T)		27.8	55.2	12.47
Adapted WER	8.5		25.9	50.9	9/27

Table 1: Characteristics of state-of-the-art models. The difficulty in completing a chart like this reflects the fact that experiments have so many dimensions that they are rarely comparable.

mother tongue. Influences include phonemic and phonological effects, word choice, and syntactic structure. Learners of English also show fluency effects, such as slower pace, disfluencies, and silence between words. All of these impact ASR performance. Approaches to accented ASR historically focused on modifying elements of the linguistic model (for example, changing pronunciations in the lexicon to reflect what an accented speaker might actually say) or the acoustic model (including accented speech in training or interpolating the acoustic models after training). With DNN-based ASR, the vocabulary is usually statistically determined, and the phone set is not needed if text symbols are predicted directly. Because accented recordings are both relatively scarce and hard to verify (how do we determine what an accent is and how strong it is? How do we determine correctness of the reference transcript?), most training-based approaches introduce accented data to an established model where more can be accomplished with less data.

Despite attempts by many researchers using a wide variety of architectures and data, performance on accented speakers still lags behind that of "standard" speakers. We aim to address this gap by building on selected state of the art systems.

There is wide variation in the types of systems, types of speakers, and end goals of recent work. Google reports that its RNN-T model has a better baseline on L2-Arctic data than its cloud model, and that finetuning with accented data yields further improvements. They report that models vary in how much training data is needed, so this will be something to keep in mind. Amazon [5] reports gains with transfer learning and a phoneme mapping scheme that allows them to leverage a pronunciation dictionary in the accented dialect. They are comparing native dialects (US, GB, AU, IN), so there are no learner effects. It is particularly important to be clear about Indian speaker characteristics, as there are native speakers of Indian English, a fully formed variety, as well as learners from very diverse linguistic backgrounds. Multi-task learning (e.g. [17]) can be used to train a model with two goals in mind, a main task and an auxiliary task. For accented speech, the auxiliary tasks can be the individual accents. In this case, it is necessary to know what the accent is so the task can be specified during training and inference. MTL has also been applied with graphemes and phonemes as the auxiliary tasks [13]. In this case, the two will ideally inform each other. Jain et al (IIT) [9] also use multi-task learning to jointly achieve accent prediction and ASR. They report results for 7 native accents of English (e.g. US, AU, Welsh) Shibano et al (UBC) [14] report promising results using accent-specific fine tuning on a wav2vec model using only a small amount of training data. They compare different configurations of speaker/accent dependency and find that different accents respond to different approaches.

3 Model Description

3.1 Off-the-shelf model

The model used for Off-the-shelf model here is a HuggingFace implementation [8] of Whisper, an off-the-shelf, multilingual speech recognition model trained by OpenAI. To improve training and inference times, we choose Whisper-small. Whisper has a 30 second limit on audio input length. To accommodate the long utterances in LibriSpeech, we will segment the audio into shorter sub-utterances at sentence boundaries.

Whisper outputs character-level transcriptions. We will calculate word error rate (WER) between the predicted transcriptions and ground truth to benchmark performance.

3.2 Evaluation Metric

Word error rate (WER) is a common metric used to measure the performance of speech recognition and machine translation models. It is a metric derived from Levenshtein distance on the word level. To compute the metric we align the predictions with reference labels, and then the rate can be computed as

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

where S, D, I are the number of substitutions, deletions, and insertions, respectively. C is the number of correct words.

3.3 AccentDB classifier and Fine-tuned models

The model for accented speech is based on the AccentDB classifier and Whisper's encoder architecture. [7] The accent classification component is a 1D convolutional neural network (CNN) trained to predict accent labels, with 3 layers of Conv1d, Maxpool, and ReLU before 2 dense layers, dropout, and softmax.For accent neutralization, we construct MFCC features through the Whisper feature extractor, or in the case of AccentDB, the python_speech_features library. Then we train/fine-tune an ASR model for each accent, where the encoder derives neutral features from the accented input. This can be used to reconstruct the input transcript.

During inference, the accent classifier predicts the accent of the input audio. The corresponding Whisper model is applied to neutralize the accent, performing speech recognition/transcription. We will calculate WER between the predicted transcriptions on original vs. neutralized audio to evaluate the accent normalization approach. Phoneme error rate (PER) will also be used to analyze misrecognized sounds.

3.3.1 Loss function

We used the same cross-entropy loss function used by the AccentDB authors with ADAM optimizer and learning rate of 0.001. We used the same special Word Error Rate defined in Whisper which minimizes penalisation due to non-semantics level differences as explained in [2].

The cross-entropy function is calculated as follows:

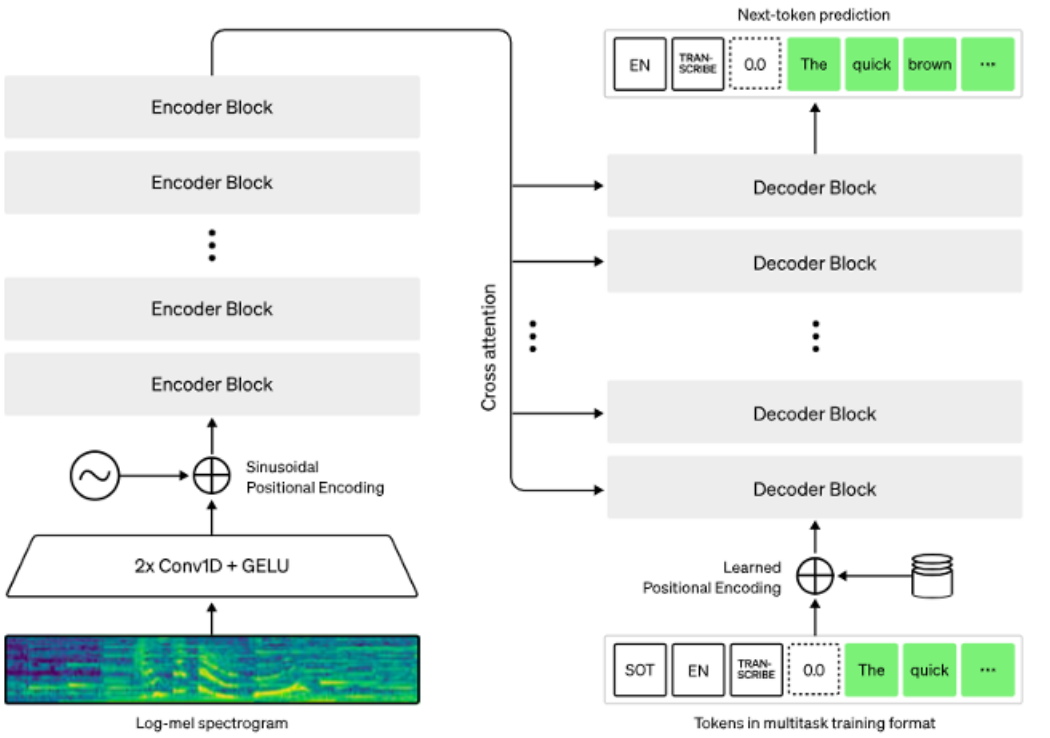
$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Where M is the size of classification, $\log$ is the natural log, y is the indicator for correct classification and p is the probability that observation o is predicted as c.

In our case, we are classifying our data into 4 different Indian accents, so M should be 4, and o and c could be Bangla, Malayalam, Odiya, or Telugu.

3.3.2 Whisper

The Whisper architecture for the model we used is described in the below diagram.



3.3.3 Accent Classification

Architecture
Utilized 1-D Conv Network from https://accentdb.org/ composed of 3-1 D convolution layer followed by a 2 fully connected layers and Softmax layer to compute class probability (duplicated published data).

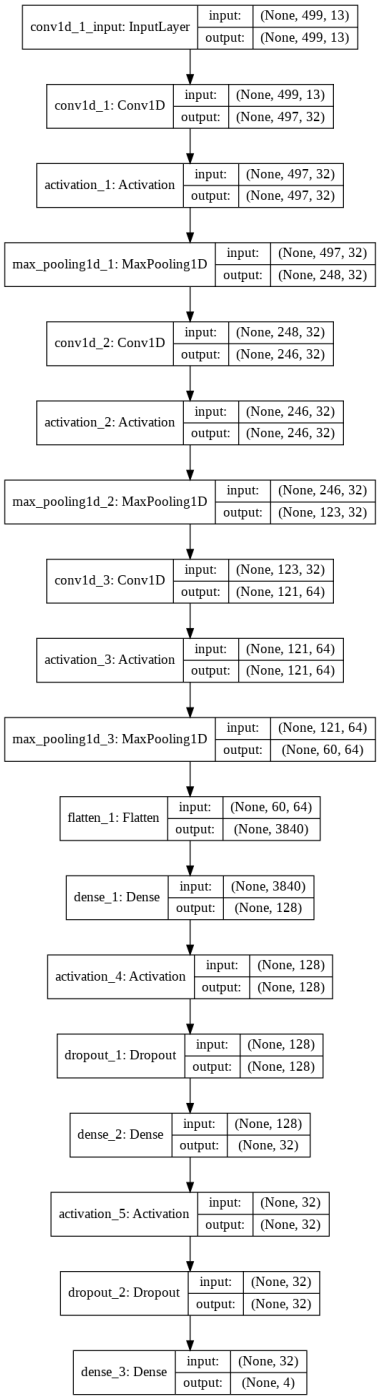
Input to model
(n, 499, 13)

n = number of input files (5 sec each)

499 - 10 msec segments with 1 msed overlap

13 Power-Frequency features

The below diagram describes the Classification layer architecture.



4 Datasets

4.1 Data sources

Data comes from the Speech Accent Archive[16], AccentDB[3], and Commonvoice[4].

4.1.1 Speech Accent Archive

This corpus was collected with an aim to provide detailed linguistic information about a variety of accents. Speakers all read the same phonetically balanced 4-sentence text. There is exactly one

recording per speaker, with duration between approximately 15-40 seconds depending on fluency. A variety of native English accents are recorded; these speakers are excluded from our work.

Native English speakers: 1561
Non-native English speakers: 579
Accents: 202 (including "English," which covers regional variation)
Indian-accented: 93
Indian varieties represented: 12
Speech type: Read
Prompt source: Phonetically balanced, custom, one sentence
Dataset size: 1561 recordings (non-native) / 579 recordings (native English)
Sampling rate: 44.1kHz

4.1.2 AccentDB

The AccentDB project focuses on Indian-accented English. Speakers of 4 mother tongues were invited to record phonetically balanced sentences in a laboratory setting. Native English recordings are also provided, but these are synthetic speech generated by Amazon Polly. At the time of writing, recordings had not been provided; only sample vectors were available.

Native English speakers: 15 (synthetic)
Accented, non-Indian: N/A
Indian-accented: 8
Indian varieties represented: 4
Speech type: Read
Prompt source: Harvard sentences [1]
Dataset size: 9 hours (Indian) / 10.75 hours (synthetic Native)
Sampling rate: 16kHz, downsampled to 8kHz during MFCC extraction

4.1.3 Commonvoice

The Commonvoice corpus project invites volunteers to donate speech samples. Data is collected and validated through crowdsourcing. Because of the complexity of characterizing and documenting accent, the latest versions of the database do not include an accent field. We are therefore using version 5.1 for our experiments. 49% of speakers in the v5.1 English dataset self-reported an accent.

Native English accents: 42%
Accented, non-Indian: 2%
Indian-accented: 5%
Speech type: Read
Prompt source: Wikipedia
Speakers: 61528
Dataset size: 1470 hours (validated)
Sampling rate: 48kHz

4.1.4 Working with the Data

The CommonVoice data is well-documented and available through HuggingFace. Working with v5.1 as described above, we extracted train and test splits, and filtered out utterances with an Indian accent (identified by the speaker, in the "accent" column of the dataset). This data is downsampled to 16kHz and passed to the Whisper feature extractor, which derives a log power melspectrogram, with 80 features and padded to 30s. The labels, or transcripts, are processed by the Whisper tokenizer. Both of these objects have pre-trained weights/filters which we load from HuggingFace.

We treated Speech Accent Archive as a custom dataset, downloading it, labelling each sample with and organizing it into a HuggingFace "audiofolder" format. This can then be loaded, downsampled, and preprocessed similarly to CommonVoice.

The AccentDB package only provides sample vector files, without transcriptions. Quite a bit of work was done tracing the source code and comparing it to librosa audio utils, and the WhisperFeatureExtractor. We determined that speech2vec and mfcc.py in AccentDB uses the python_speech_features

package to extract MFCCs from raw audio. This feature extraction is greatly simplified and differs from Whisper:

- audio is down sampled to 8kHz instead of 16
- 26 "mels", or filters, are used for the melspectrogram, along with natural log vs log10
- the DFT analysis window and overlap is smaller (10:1ms vs 25:10ms) [3]
- no window fx is used, while Whisper uses the default Hann window
- 13 MFCCs (26, then truncate) are extracted from the spectrogram, through a DCT and cepstral lifting

Following our analysis we tried a couple techniques to match the Whisper input. Mainly, we tried reconstructing the raw audio via librosa.feature.inverse.mfcc_to_audio. This is a bit wasteful as it undoes the preprocessing of the AccentDB team, but it allows WhisperFeatureExtractor to create its own well-formatted spectrogram. Unfortunately we could not verify the effectiveness as sentence labellings did not work as expected. Although the utterances are all from the Harvard sentence set, it was not possible to identify which vector goes with which sentence; some sentences are excluded while others duplicated.

Another idea to make use of the preprocessed AccentDB data is to prepend a channel expansion layer (Conv2d) in the pretrained Whisper model, from 13 to 80 mels. The weights for this layer would have to be trained from scratch, but if it works we can forgo the lossy, error-prone process of reconstructing audio. As this was a major blocker for our project we reached out to the AccentDB authors for the original wav files and transcript labels, which we hope to obtain before the winter break.

5 Baseline Selection

5.1 Off-the-shelf models

To establish baseline expectations for ASR performance on non-native speech, we tested the base Whisper [12] model on the SpeechArchive data set. After approximating the reported result[11] of 4.26% WER on the LibriSpeech test-clean data set, we established a baseline on the SpeechArchive set of non-native accented data of 11.45% WER.

5.2 AccentDB models

Comparing the codebases found on the Internet, we chose to work on the model from [3] to build a baseline. AccentDB provides a dataset of clean, manually-labeled data in Indian accent. The paper also provides analysis on the separability of data, which further demonstrate the quality of the dataset. Meanshile, the results posted in [3] seem to provide high accuracy in classification tasks. [3] added accent neutralization with an autoregressive encoder, and this step also showed a high accuracy. As a well-established, recent dataset with public dataset and codebase, we believe it could be a good choice as our baseline model.

[3] provides a codebase with the classifier in accented data, and we believe the code could be a good start to validate their data and build our own baseline model. Our target is to generalize the model and the classifier to a wider range of data available online, and our main task is to implement better accent neutralization methods to achieve generalization.

6 Baseline Implementation Completeness

The first step in implementing accent neutralization is recognizing and classifying the accent. Once the accent in the input speech is recognized, the second step would be to map this accent to a network that can regenerate the accent free speech. Our baseline is from [3] where the authors collected accented data sets that is manually labeled. The classifier uses a either an MLP or 1-D convolution or 1-D convolution with attention but does not seem to be any significant difference beyond 0.5% difference among any of these classifiers. The attention approach described in the paper tries to augment the classification by analyzing the audio segments that can of significantly important for the classification model but the details are not clear enough. The second step of accent neutralization

uses transformation models as an inference-time pre-processing for ASR systems. It uses a stacked denoising autoencoder architecture made of a series of convolutional and pooling layers followed by de-convolutional and pooling layers with the tanch activation at each layer output. The convolution layers act as feature extractors of input MFCC encoding them into dense representations. The de-convolution layers learn the transformations on this dense representation for reconstructing MFCC features of the accent free target. Tow of these autoencoder networks are trained, one is trained with accent MFCC as input with accent free MFCC as target ouput while the second encoder is trained with the swapped pairs. Mean-square error between between input and output vectors is a feature-wise loss function of the autoencoder. RMSProp optimizer is used for trainng with 0.01 learning rate.

6.1 Off-the-shelf models

The accent classification network is a 1D convolution network that comprises three 1-D convolution layer with Relu activation and MaxPooling. The convolution layers are followed by two fully connected layers with Relu activation followed by Dropout layer. The output layer is a fully connected layer with Softmax activation to compute the probabilities of each of the accent classes and then cross entropy loss is computed for the network. The model uses Adam optimized with 0.01 learning rate. We modified the published colab convolution network classifier code to use the colab storage with the need to access the perosnal google drive and validated the published training results showing more than 90% training accuracy.
To complete our baseline model investigation, we need to do the following:

- Segment utterances at the sentence level so that all SpeechArchive utterances can be included
- Generate phoneme-level output so that patterns in misrecognized phonemes can be identified
- Test OTS models other than Whisper
- Test L2 data other than SpeechArchive so that a comparison to at least one of the results from Table 1 can be provided. We hope to test both L2-Arctic and Commonvoice.
- Test the AccentDB classifier on a more diverse data set to quantify the generalization error.
- Implement the accent neutralization autoencoder either by starting with baseline version obtained from the authors of [3] or do our own implimentation from scratch.

7 Implemented Extensions

The final goal is to create a streamlined pipeline to take accented speech as input and generate accurate transcripts. These transcripts can be used to calculate metrics like speech accuracy, phoneme conflicts, and points of improvement for the accented input (which can be leveraged for language learning). Our proposed extension is to finetune existing ASR models with region-specific accented English – for our experiments, we focused on Indian/Hindi. While models like Whisper are multilingual, there is no support for specific accents. Our second extension is the integration of AccentDB's accent classifier. This allows us to use a specialized model for each accent, without the need for the user to input their own accent. As detailed in Piazza followup discussion, one can imagine the deployment of a region or "accent category" (e.g. Indian, European) as separate backend services a system like Siri can access.

We further split our goal of refining the models and datasets into 2 parts: Off-the-shelf and AccentDB components.

7.1 Off-the-shelf

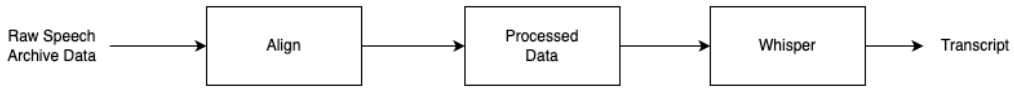
The plan is to split the SpeechArchive paragraph based on utterances or sentences which fall within the 30 second hard limit for using whisper. Then we can provide more accurate insights into the performance of Whisper on SpeechArchive. We do not extend the Whisper model, but going forward we should experiment with different sizes and potentially add a channel expansion layer for AcccentDB data.
With time, we also want to look at the performance of other latest ASR off-the-shelf models, like CMU ESPNet, and SpeechBrain on SPeechArchive data based on the bandwidth we have as a team.

Model	Test set	WER
Whisper-base (baseline)	LibriSpeech	4.28
Whisper-base	SpeechArchive-all	11.45
Whisper-base	SpeechArchive-Indian	9.0
Whisper-medium	SpeechArchive-all	8.43
Whisper-medium	SpeechArchive-Indian	5.24
Whisper-small	SpeechArchive-all	5.28
Whisper-small + SpeechArchive FT	SpeechArchive-Hindi	1.45

Table 2: Fine-tuning a pretrained Whisper model on Speech Accent Archive data

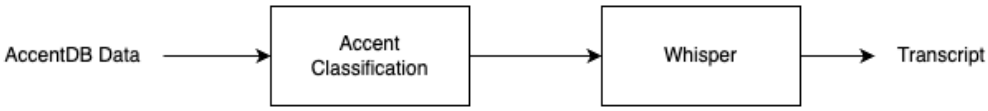
Step	Training Loss	Validation Loss	WER
1000	0.327300	0.357067	20.558631
2000	0.173400	0.341527	19.124308

Table 3: Partial results from fine-tuning whisper-small with Commonvoice data



7.2 AccentDB

Here, we classified the MFCCs extracted from the audio, then use Whisper to generate the transcripts. With a strong enough model we can provide insights on the speech accuracy of a speaker, to create data points where the input speech can be refined from a second language teachers perspective. There is also the challenge of refining the preprocessing done by the AccentDB team, extracting a log melspectrogram for Whisper's use.



8 Results

As an entry point to understanding how fine-tuning with accented data can impact ASR performance on accented speech, we tested several configurations of the Speech Accent Archive data set applied to a Whisper pretrained model. Results are shown in Table 8. The Whisper-small+SpeechArchiveFT model used the whisper-small pretrained model and fine-tuned using the non-Indian accented speech data, with Indian-accented speech used for the eval development set. Hindi-accented data was held out and used for testing. The resulting improvement in WER is a positive sign, but it is important to note that 1) this is a very small data set and 2) the text of all utterances is the same, so the likelihood of overtraining must be kept in mind. Nonetheless, these results indicate that the general approach has promise.

Due to connectivity issues we were not successful in completing a model applying Commonvoice Indian-accented data to fine-tuning of whisper-small. Results shown in Table 8, showing a decrease in WER through training iterations, suggest that this approach will yield positive results for the larger dataset (which should be less susceptible to overfitting).

9 Conclusion and Future Work

9.1 Conclusion

The core problem this work addresses is performance disparities in automatic speech recognition (ASR) systems for non-native speakers. The goal is creating accent-adapted models that can interpret diverse speech patterns to enable equitable access to language learning tools.

We have made progress towards a pipeline for accent-adapted ASR to support language learning. The accent classification stage was successfully implemented, matching published accuracy levels. However, challenges remain in preparing compatible accent speech data to enable effective fine-tuning of the Whisper model. We are currently working withe AccentDB authors to retrieve the row audio files required as the input to fine tune Whisper as well as the text aligned transcripts to train the Whisper Decoder. The authors responded today committing to provide the files so that we can perform the Whisper fine tuning.

Once the full pipeline is complete, it has the potential to facilitate personalized learning and pronunciation feedback for non-native speakers. Fine-tuned ASR models adapted to specific accents can better interpret speech patterns that differ from native speakers. Comparing recognition results between original accented speech and neutralized versions would reveal areas needing improvement.

9.2 Future Work

Our next steps should focus on reconstructing usable audio waveforms from the AccentDB Mel-frequency cepstrum coefficients. With compatible data in place, multiple experiments can compare ASR accuracy with and without accent neutralization. The sensitivity of performance gains to factors like accent diversity, training hours, etc. should be analyzed as well. We would like to experiment with our pipeline that is trained to neutralize Indian specific accents to English to other languages like Arabic. Arabic is a language that has multiple dialects and would be useful to have a tool that enables feedback for teaching proper reading and written comprehension. //

Additional priority areas include

1. Testing latest ASR models like ESPnet and SpeechBrain on multi-accent datasets
2. Comparing individual vs. pooled accent adaptation approaches
3. Leveraging unlabeled accent data in semi-supervised learning
4. Formulating personalized strategies linked to ASR corrections to assist language learners
5. Fulfilling these next steps will lead to an end-to-end system that can democratize modern speech recognition technology for non-native speakers. The insights on accent-related intelligibility challenges can make language resources more accessible globally.
6. Turn this project into a functional paper, which we plan on publishing early next year.

10 Acknowledgments

We would to acknowledge the AccentDB authors Afroz Ahamad and Pranesh Bhargava for corresponding with the team and committing to provide the audio data and the text aligned transcripts needed for the future work. We would like to acknowledge Dr. Laura Tomokiyo who contributed to the project at the level of all authors and providing her expertise on speech and education even she is not taking the class for credit.

11 Division of Work

Github

- `https://github.com/aliu39/idl-accented-asr`

Harsha Tummala

- Implement accent classification model
- Prepare AccentDB dataset
- Reconstruct audio from MFCC features
- Write final paper and create summary slides

Andrew Liu

- Literature review on accent adaptation techniques
- Prepare Speech Accent Archive dataset
- Compare ASR performance - original vs neutralized
- Analyze phoneme error patterns

Yitong Chen

- Benchmark latest ASR models on multi-accent data
- Experiment with individual vs pooled accent adaptation
- Formulate learning strategies from ASR corrections

Mohamed Abdelmoneum

- Fine-tune Whisper ASR model on accents
- Implement optimizations to accent neutralization
- Semi-supervised learning with unlabeled accent data
- Evaluate accent classification accuracy

References

[1] Ieee recommended practice for speech quality measurements. *IEEE No 297-1969*, pages 1–24, 1969.

[2] T. Xu C. McLeavey I. sutskever A. Radford, J. Kim. Robust speech recognition via large-scale weak supervision.

[3] Ankit Anand Afroz Ahamad and Pranesh Bhargava. Accentdb: A database of non-native english accent to assist neural speech recognition. *arXiv preprint arXiv:2005.07973*, 2020.

[4] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber. Common voice: A massively-multilingual speech corpus. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 4211–4215, 2020.

[5] Harish Arsikere, Ashtosh Sapru, and Sri Garimella. Multi-dialect acoustic modeling using phone mapping and online i-vectors. 2019.

[6] Yi-Chen Chen, Zhaojun Yang, Ching-Feng Yeh, Mahaveer Jain, and Michael L. Seltzer. Aipnet: Generative adversarial pre-training of accent-invariant networks for end-to-end speech recognition, 2019.

[7] Team Implementation. Accentdb implementation.

[8] Team Implementation. Off-the-shelf model code.

[9] Abhinav Jain, Minali Upreti, and Preethi Jyothi. Improved accented speech recognition using accent embeddings and multi-task learning. In *Interspeech*, pages 2454–2458, 2018.

[10] Nina Markl and Catherine Lai. Everyone has an accent. In *Interspeech 2023*. ISCA, 2023.

[11] OpenAI.

[12] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[13] Kanishka Rao and Haşim Sak. Multi-accent speech recognition with hierarchical grapheme based models. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4815–4819. IEEE, 2017.

[14] Toshiko Shibano, Xinyi Zhang, Mia Taige Li, Haejin Cho, Peter Sullivan, and Muhammad Abdul-Mageed. Speech technology for everyone: Automatic speech recognition for non-native english with transfer learning. *arXiv preprint arXiv:2110.00678*, 2021.

[15] Joel Shor, Dotan Emanuel, Oran Lang, Omry Tuval, Michael Brenner, Julie Cattiau, Fernando Vieira, Maeve McNally, Taylor Charbonneau, Melissa Nollstadt, et al. Personalizing asr for dysarthric and accented speech with limited data. *arXiv preprint arXiv:1907.13511*, 2019.

[16] Steven Weinberger. Speech accent archive, 2015.

[17] Xuesong Yang, Kartik Audhkhasi, Andrew Rosenberg, Samuel Thomas, Bhuvana Ramabhadran, and Mark Hasegawa-Johnson. Joint modeling of accents and acoustics for multi-accent speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2018.