SPEECH-TO-TEXT MODELING USING FEATURE-WISE INTERACTIONS BETWEEN SPEECH, GENDER, AND ACCENT

Ankit Mathur anmath@iu.edu
Saurabh Mathur samathur@iu.edu

Indiana University Bloomington

ABSTRACT

This project implements a deep-learning based speech-to-text model for the English language that combines the information about speaker's gender and accent using conditioning mechanisms. In order to achieve this, we built two separate models:

1. A classification model that uses deep learning to predict the gender and accent of the speaker

2. A seq-2-seq model that uses deep learning to convert speech into text

We then used a conditioning layer[3] that combines these two neural networks such that the final output is conditioned on the features learned by both the neural networks. Finally, we evaluated the results for each gender and accent combination using the CTC loss function (i.e. soft-edit distance) as the evaluation metric.

For the baseline, we've used the DeepSpeech model[1, 2] that uses deep-learning to perform end-to-end speech recognition. However, since DeepSpeech requires different models for different accents, it suffers from scalability issues. Hence, this project aims to develop a single model that adapts itself based upon the speaker's accent.

1. INTRODUCTION

An Automatic Speech Recognition (ASR) system converts spoken words into computer-readable text. ASR systems have a wide variety of applications from electronic personal assistants to subtitle generation systems. While ability to recognize speech automatically has increased dramatically over the past due to deep neural networks, ASR systems still need separate models for different accents. This approach is not very scalable. In this project, we attempt to develop a single model that can adapt itself according to the accent of the speaker.

2. DATASETS

In our project, we used three publicly available and tagged datasets for experimentation purposes:

1. The **Speech Accent Archive** by George Mason University presents $1K$ audio samples totaling to over 5 hours of speech data. These samples were spoken by $1K$ native and non-native English speakers distributed over 10 combinations of accent and gender, which were used as the labels for our classification model. Accent-Gender combinations in the GMU Speech Accent Archive:

accent	count
'male-usa-english'	189
'female-usa-english'	184
'male-uk-english'	41
'female-china-mandarin'	33
'male-canada-english'	27
'female-south korea-korean'	27
'female-uk-english'	24
'male-south korea-korean'	23
'male-saudi arabia-arabic'	22
'male-italy-italian'	22

2. The **DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus**[6] consists of $6.3K$ audio samples, which is equivalent to around 10 hours of speech. They were spoken by 630 speakers (10 samples per speaker) distributed over 2 genders and 8 dialects. We used this dataset for experimenting with our seq-2-seq model.

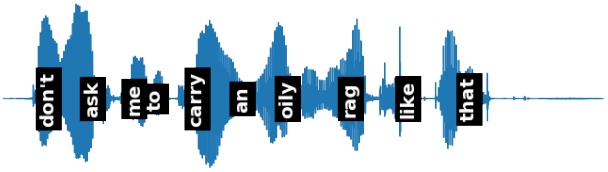

Fig. 1. An example from the TIMIT corpus

3. The **Mozilla Common Voice** dataset, which is our main dataset for this project, comprises of more than $700K$ audio samples amounting to more than $1K$ hours of speech signals. More than $41K$ speakers split across 2 genders and 17 accents contributed in making this huge dataset.

Given the size of our main dataset, we laid out a series

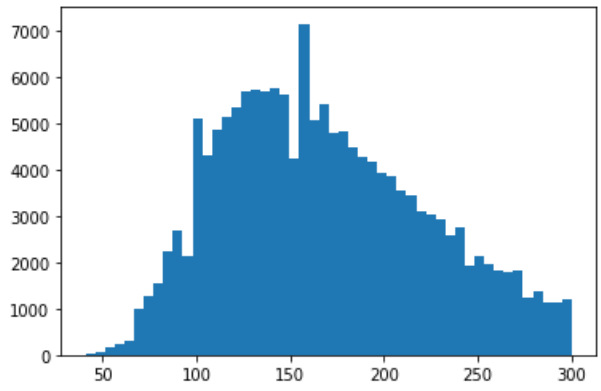

Fig. 2. Length of audio clips in Common Voice

of data pre-processing steps to extract *clean and relevant* speech signals from the Mozilla Common Voice dataset for our model.

2.1. Data Pre-processing

As one would expect, not all 17 accents in the Common Voice dataset have enough speech signals for us to train a deep neural network on. Hence, in this project, we focused on accents from the following 6 countries:

1. USA
2. Canada
3. Australia
4. England
5. Scotland
6. India

The table below showcases the Accent-Gender combinations chosen from the Common Voice dataset:

accent	count
('australia', 'female')	2167
('australia', 'male')	10453
('canada', 'female')	4833
('canada', 'male')	8226
('england', 'female')	4319
('england', 'male')	33420
('indian', 'female')	5752
('indian', 'male')	12507
('scotland', 'female')	1032
('scotland', 'male')	2590
('us', 'female')	20493
('us', 'male')	70445

Furthermore, since a lot of speakers did not have labeled gender and accent related information, we subsetted the original dataset to include a total of $5K$ speakers and $262K$ audio signals that are worth about 360 hours of speech data.

The signals thus identified were then split into train, validation, and test sets by dividing the speakers into 70%, 15%, and 15% respectively such that no two sets have any common speakers for a given combination of gender and accent. The final train, validation, and test sets thus obtained had $176K$, $41K$, $45K$ audio signals respectively, which were then used in the feature extraction process.

While the TIMIT dataset was created in a controlled environment and using the same recording equipment, the Common Voice dataset has been created in a decentralized manner by volunteers around the globe. As a result, the common voice dataset is much more noisy and has a higher variance in the audio samples.

2.2. Feature Extraction

In order to extract features from the audio signals, we applied both **STFT** and **MFCC** to the Common Voice data. Since we had more than $250K$ audio samples in total, carrying out the feature extraction process on local machine or *Google Colab* was out of the question. Hence, we established a feature extraction pipeline that comprised of the following steps:

1. Set up the Deep Learning VM capable of carrying out heavy computation using the Google Compute Engine. We used the one with 1 Nvidia V100 GPU, 8 vCPUs, 64GB RAM, and 300GB of SSD storage.

2. Transfer the zipped audio files from Google Drive to Google Cloud bucket. This was done using Google Colab and took about 10 minutes in total.

3. SSH into the VM and transfer the files from Google Cloud bucket to VM (10 minutes).

4. Run the script that applies STFT and MFCC transformations on a chunk of audio files. Since, the script can be setup to process a fixed set of audio fiels, many instances of the script can be run in parallel. For this project, we ran about 10 parallel instances on the VM, each working on a batch of $5K$ files. This was computationally the most expensive step and took about 8 hours of continuous run-time on the Deep Learning VM.

5. Zip the files on VM and transfer them back to Google Cloud bucket (20 minutes).

6. Using Colab connected to Drive, transfer the transformed and zipped audio files back to Drive where they can be accessed for modeling purposes.

3. MODEL ARCHITECTURE

Since the Long Short-Term Memory (LSTM) neural networks can have both input and output as a sequence, we used LSTM based models for both tasks. Since each audio has a different length, we padded each batch to the length of the longest sequence in the batch.

3.1. Accent classification model

For the accent classification task, we used a two-layer LSTM followed by a fully-connected layer. The output of the LSTM layers pooled over time by max pooling and average pooling. The resulting concatenated vector is fed to the fully-connected layer.

The softmax activation function is the last layer in deep neural network. It is helpful because it specifies a discrete probability distribution for K classes, denoted by $\Sigma_{k=1}^{K} p_k$.

Since, both datasets (GMU Speech accent archive and Common voice) were unbalanced, we used a class weighted cross-entropy loss function to train this model. The weight for each class is inversely proportional to its frequency in the training data.

$$weight_{c_i} \propto \frac{1}{freq(c_i)}$$

3.2. Speech recognition model

For the speech recognition task, we used a similar network having a two-layer LSTM followed by a fully-connected layer. However, instead of pooling the output of the LSTM layers, the fully-connected layers are applied at each timestep. Further, we take log-softmax at each timestep. As a result, we get a discrete distribution over all characters at each timestep.

To learn the weights of this neural network we used the connectionist temporal classification loss (CTC). This loss function is used for alignment-free prediction i.e. this loss enables us to predict an entire sentence without having the split up the audio into pieces. Heuristically, the CTC loss computes a soft edit distance between the predicted sequence and the ground truth. Our implementation uses the CTCLoss class provided by PyTorch.

$$weight(c_i) \propto \frac{1}{freq(c_i)}$$

The softmax activation function is the last layer in deep neural network. It is helpful because it specifies a discrete probability distribution for K classes, denoted by $\Sigma_{k=1}^{K} p_k$.

3.3. Conditioned Speech Recognition model

The TIMIT dataset and the Common Voice dataset both provide information about the gender and the accent of the speaker. So, We use this information to condition our speech recognition neural network. A neural network without conditioning would predict:

$$P(char_i|MFCC, char_0, ..., char_{i-1})$$

The conditioned neural network now predicts

$$P(char_i|MFCC, char_0, ..., char_{i-1}, Gender, Accent)$$

. This extra information is encoded as a one hot vector and concatenated with the feature vector at each timestep.

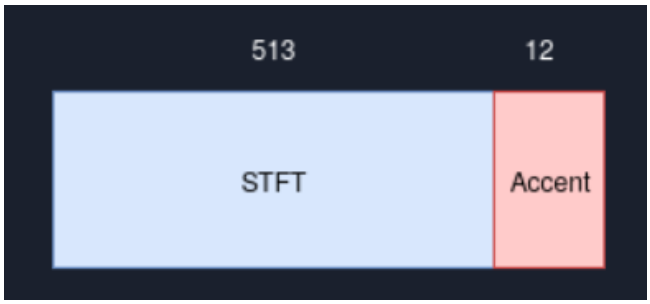

Fig. 3. Conditioning mechanism using feature concatenation

3.4. CTC loss

The CTC algorithm[4] is an alignment-free algorithm that uses a *blank* token ϵ to align the spoken words and the output text as shown in the figure below.

Fig. 4. Working of the CTC algorithm using *blank* tokens

Mathematically, the CTC conditional probability is computed as shown in the equation below:

4. EXPERIMENTS

To evaluate the performance of the models, we employed the following metrics:

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability** **marginalizes** over the set of valid alignments computing the **probability** for a single alignment step-by-step.

1. CTC Loss: Soft edit distance between prediction and ground truth

2. Test Accuracy: The trained model performance on unseen data.

3. Confusion Matrix: The table for describing classification performance.

4.1. Speech Accent Archive

As mentioned above, we used this dataset for building a *gender* × *accent* classification model. In order to achieve this task, we implemented a two-layered LSTM model each with 128 units, as shown in figure below, on the audio signals transformed using STFT. Since we had 10 combinations of *gender* × *accent*, the output of the model has 10 classes as shown in the figure.

The confusion matrix of the accent classification model is also showcased in the figure below. As we can see, the model performed really well on this data set.

```
Net(
  (lstm): LSTM(513, 128, num_layers=2, batch_first=True)
  (linear): Linear(in_features=256, out_features=10, bias=True)
)
```

Fig. 5. Model used for Accent classification on the Speech Accent Archive

4.2. DARPA TIMIT Speech Corpus

We used this dataset for implementing the speech-2-text model. The model, yet again, had two layers of LSTM each containing 128 units. Since this is a seq-2-seq model, the shape of the output was the same as that of the input.

We used Adam optimizer with weight decay to train this model and minimized the Connectionist Temporal Classification(CTC) loss. In order to tackle the gradient explosion, the gradient norm was clipped at 400.

This model was implemented both with and without concatenating the one-hot inputs for *gender* and *accent*. The learning curves for both implementations are shown in the figures below. As we can see, both train and test losses seem

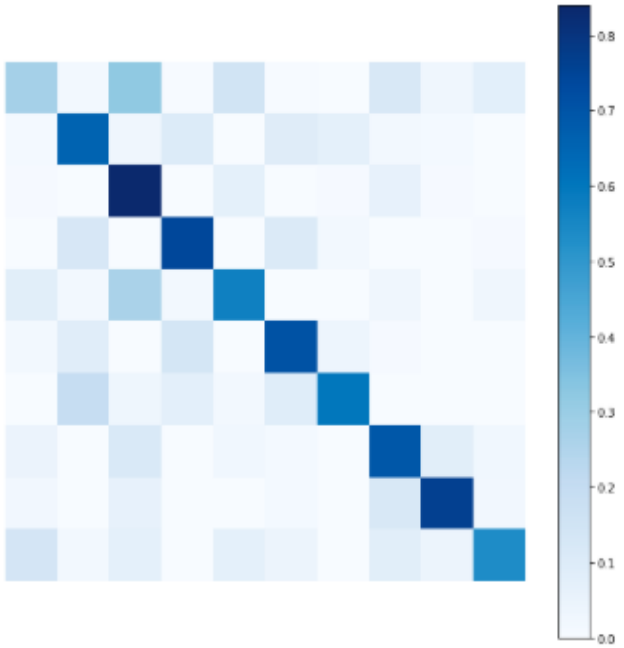

Fig. 6. Confusion matrix of the accent classification model

to decrease when using the conditioning mechanism. Furthermore, the test loss tends to converge more when using conditioning vs. without conditioning.

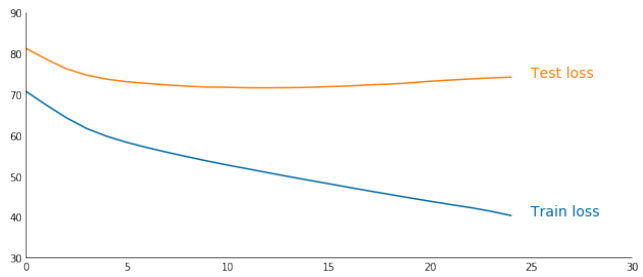

Fig. 7. Learning curve of the speech-2-text model without conditioning

4.3. Mozilla Common Voice

The model architecture and experimental setup for this dataset is same as with the TIMIT dataset. The only difference was that we used MFCC features instead of STFT features primarily because MFCC provides much lesser number of features, thereby dealing with the scalability issues. Despite that, each training epoch over this dataset, running the GPU-enabled VM described previously, took about 30 minutes to complete. Hence, due to time constraints, we ran 10 epochs of this model and observed that the results were pretty pho-

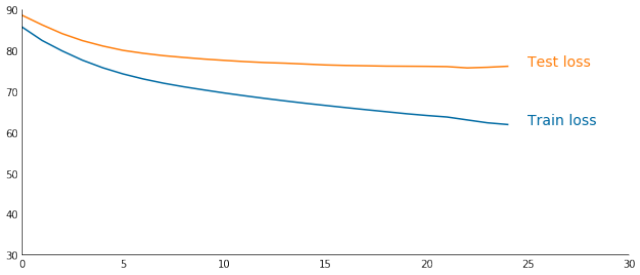

Fig. 8. Learning curve of the speech-2-text model with conditioning

netic in nature, but lacked the language structure, as has been showcased below:

Ground Truth:	*destroy every file related to my audits*
Prediction:	*es proy evreyfo rtlathemotes*
Ground Truth:	*nobody in his right mind punishes a quarter century old dereliction*
Prediction:	*non melit ias righ moting ciishous if alervershantre dar lit h*
Ground Truth:	*the cranberry bog gets very pretty in autumn*
Prediction:	*he crn brid bo dis puri thr e a or*
Ground Truth:	*don't ask me to carry an oily rag like that*
Prediction:	*don't ask me to cacarry an oily rag like that*
Ground Truth:	*she had your dark suit in greasy wash water all year*
Prediction:	*she had your dark suit in greasy wash water all year*

Fig. 9. Sample output of the speech-2-text mode

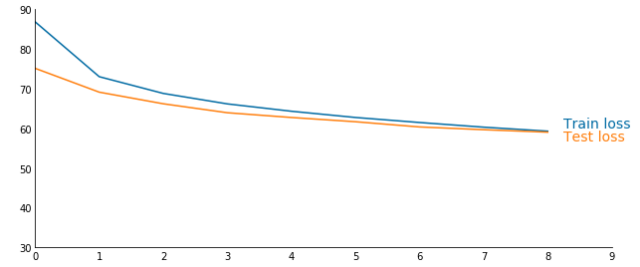

Fig. 10. Learning curve of the speech recognition model on Mozilla Common Voice

Sample output:
GT: a dish fit for the gods
P: a dish ti fhe thegads
GT: when she entered the dance floor everyone in the nightclub was looking at her
P: whe she handte the dans fo avery wie an the niclob was loofing anver GT: oh is that al¿
P: il is that all

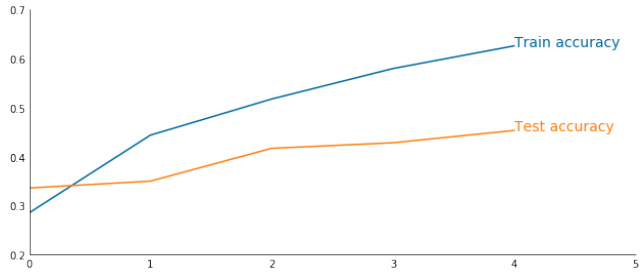

Fig. 11. Learning curve of the accent classification model on Mozilla Common Voice

5. CONCLUSION AND FUTURE WORK

In this project, we have tried to build a unified ASR model for 12 accent-gender combinations. We have found that while a sufficiently large neural network can generalize over accents, adding the accent information explicitly improves generalization.

- The datasets only have labels for male/female audio samples.
- The common voice dataset is noisy. A noise-aware model might be able to generalize better.
- Due to budget and time constraints, we could not use the full DeepSpeech model. Mozilla used 8 GPUs to train the DeepSpeech model on an earlier version of the Common Voice dataset which was roughly half the current version. Clearly, we didn't have that kind of computational capacity. However, it would be interesting to run these experiments on a larger neural network and compare the boost in performance due to conditioning.

6. REFERENCES

[1] Hannun, Awni, et al. "Deep speech: Scaling up end-to-end speech recognition." *arXiv preprint arXiv:1412.5567* (2014).

[2] Amodei, Dario, et al. "Deep speech 2: End-to-end speech recognition in english and mandarin." *International conference on machine learning.* 2016.

[3] Perez, Ethan, et al. "Film: Visual reasoning with a general conditioning layer." *Thirty-Second AAAI Conference on Artificial Intelligence.* 2018.

[4] Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks Graves, A., Fernandez, S., Gomez, F. and Schmidhuber, J., 2006. *Proceedings of the 23rd international conference on Machine Learning*, pp. 369–376. DOI: 10.1145/1143844.1143891

[5] Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." *International Conference on Machine Learning*. 2016.

[6] Garofolo, John S. et al. "DARPA TIMIT: : acoustic-phonetic continuous speech corpus CD-ROM, NIST speech disc 1-1.1." (1993).