EECS 498/598 WN 2021 Project Proposal

Siwei Wang

TOTAL POINTS

(not graded)

QUESTION 1

1 Executive Summary

- 0 pts Correct

QUESTION 2

2 High-Level Description

- 0 pts Correct

QUESTION 3

3 Implementation and Issues

- 0 pts Correct

QUESTION 4

4 Ethical Considerations

- 0 pts Correct

QUESTION 5

5 Way Forward and Schedule

- 0 pts Correct

QUESTION 6

6 Final Presentation

- 0 pts Correct

QUESTION 7

7 Tools and Materials

- 0 pts Correct

QUESTION 8

8 The Team

- 0 pts Correct

1 Executive Summary

You can write "I"

There are a staggering number of accents to spoken English. We'd like automatic speech recognition systems to have the property of accent invariance, working well regardless of the accent a speaker exhibits. An intermediate step towards a solution is the classification of accents, which may even lead to generative models that convert one accent into another. Successful attempts at solutions often transformed audio waveforms into MFCC spectrograms, which we will do as well. Previous works leveraged classical ML techniques such as SVM, regression, and gradient boosting. Several deep learning methods have also been attempted such as CNN and long short-term memory. Both deep NN approaches achieved respectable performance, but I believe a combination of the two would yield a more descriptive model for human speech. **My project proposal is to solve the accent problem using a CNN that feeds into a bidirectional LSTM to predict word-level classifications; then aggregate those results to predict overall accent.** My data will primarily derive from the Speech Accent Archive [9], with supplementation from the Wildcat Corpus [10].

avoid words like 'solve'

This is really well thought out.

2 High-Level Description

2.1 Background Information

If we ranked the many languages of the world by total number of speakers and by number of countries in which that language is official, then English tops both of those lists [1]. Given its common role as *lingua franca*, it should come as no surprise that there are thousands of accents and regional dialects of English. With the proliferation of speech recognition systems such as Siri, Alexa, and Cortana, we'd like to accurately convert speech to text, regardless of the speaker's accent. A solid first step to a solution is recognition of accents present in spoken English.

The result of this project will be a system where the user provides an audio snippet of a person speaking English. The system will return its best guess as to the accent of the speaking individual. In general, we are not concerned with real-time solutions, avoiding false positives/negatives, and other design-level constraints, which affords us great flexibility in our approach.

2.2 Related Work

There are several existing papers that have tried various architectures for the task of accent classification. Chen et al. [2] used SVM, naïve bayes, and logistic regression, achieving 69.3% test accuracy. Ahmed et al. [3] introduced a novel CNN architecture they termed a "variable filter net", getting 70.3% test accuracy. Shih [4] tried softmax regression and LSTM, reaching 79.0% test accuracy. Sheng and Edmund [5] used gradient boosting, random forest, multi-layer perceptron, and CNN's to achieve 88.0% test accuracy. Although unrelated to accent classification, I'd like to highlight the command recognition model devised by Andradea et al. [6], which inspired my own solutions, as summarized in Section 2.3. Their model scored a dazzling 96.9% on the test set.

2.3 Solutions

Clearly, the popularity of deep learning has led to models that achieve greater accuracy on accent classification. From the works listed in Section 2.2, we find that two neural network models have, in particular, been successful with accent classification. The first is the LSTM, as in [4]. The second

is the CNN as in [5]. My project proposal is to combine these two approaches, similar to [6] but applied to accents and without the attention layer.

Is there a reason to drop this?

Thus, I propose the following 2 solutions to the problem of accent classification. The first is to use a BiLSTM. The second is to apply a CNN, then use the output of the CNN as input to the same (or similar) BiLSTM as in the first solution. Thus, solution 2 builds on solution 1. Both solutions will receive preprocessing wherein audio is split on words and transformed via MFCC [7, 8]. The advantages and tradeoffs for each of the 2 solutions are summarized in Table 1 below.

Table 1: Pros and Cons of Solutions

	BiLSTM	CNN + BiLSTM
Sequential Order Preserving	✓	✓
Resistant to Vanishing/Exploding ∇	✓	✓
Fast Training Time	✓	
Smaller Dataset OK	✓	
High Expressive Power		✓
Local Translation Invariance		✓

Could be interesting to compare to a deeper CNN, which would still capture context, but would be much faster.

Data augmentation will help by increasing variance, which may provide invariance to noise. However, there could be sparsity over cultures/accents, which augmentation cannot correct for.

How small is the data set?

Given the data augmentation steps I plan to take (see Section 3.4), the problem of a smaller dataset will be less grievous, albeit still relevant. The intention is that CNN + BiLSTM will be the "flagship" model while BiLSTM serves as an extension of an existing work that may serve as a benchmark. For practical purposes, it would make sense to limit the accents that the model classifies to the ones most prevalent in the datasets. There are, after all, hundreds of accents in the datasets and many of them have only a few samples.

3 Implementation and Issues

3.1 Data Preprocessing

The Speech Accent Archive [9], as of the writing of this proposal, contains 2937 audio samples, where each one consists of a person speaking the same 69 words. The first preprocessing step is

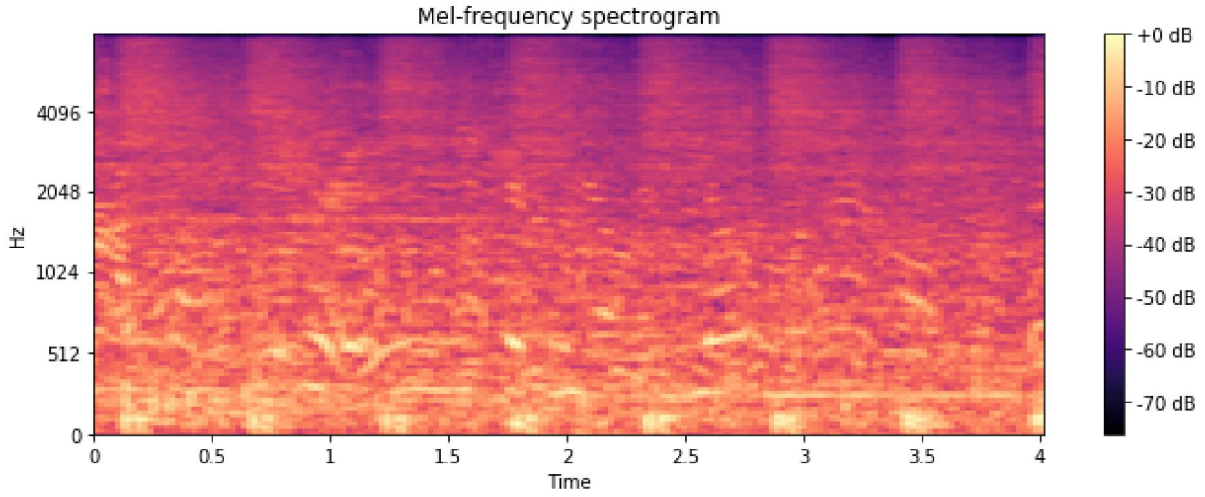
How many people are there?
How many accent classes?
May want to augment w/ other resources (e.g., Mozilla)

thus to split the audio snippet by word, which increases the size of our dataset, and negates the effect of silence between words. It is, of course, likely that length of time spent between words is a factor in determining accent, but my project will be concerned with word-level classification. Good

Audio channels are digitally represented by a stream of samples, which is just a sequence of integers. These represent the amplitude of the waveform at discrete time intervals [12]. An audio file, in turn, may have multiple channels. For example, stereo sound has 2, while "surround sound" has 6. The audio in Speech Accent Archive is stereo. We convert into mono audio by averaging the channels. You can do this in sox.

The mono-channel word utterances are then transformed into MFCC spectrograms [8]. Under the hood, this is a rather involved process of splitting the waveform into small overlapping chunks that are approximately periodic. Apply a short-time fourier transform to each chunk, convert to the mel scale, then splice the transformed chunks together. The spectrograms will then be normalized. An example of one can be seen in Figure 1 below. You can use python speech features (a python library)

Figure 1: MFCC Spectrogram [8]

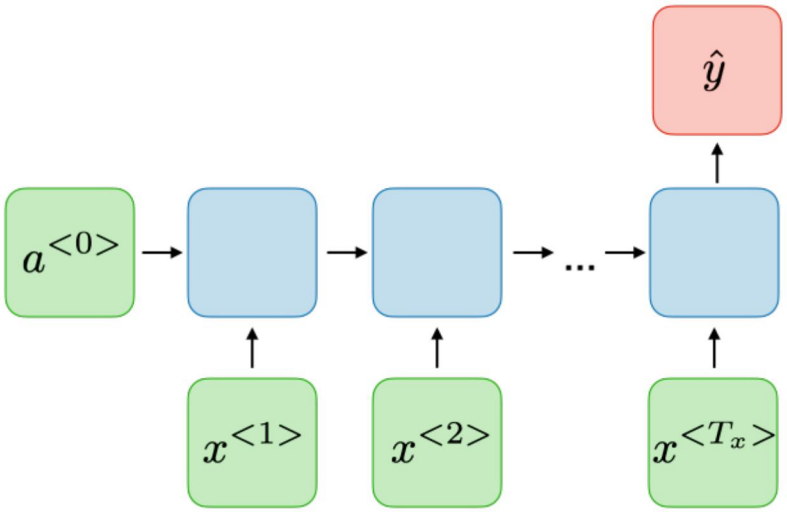


The result of preprocessing is that a 1-dimensional signal in the time domain is converted into a 2-dimensional one in the time and frequency domains. We conclude this section by introducing some mathematical notions. Let A be an audio channel with amplitude $A(t)$ at time t. We transform A into the MFCC spectrogram M with intensity $M(f, t)$ at frequency f and time t.

3.2 BiLSTM

A recurrent NN requires a series of data. In the case of the MFCC spectrogram M, the input at time t is a vector $x(t) \coloneqq (M(f_1, t), \ldots, M(f_k, t))$, where the $f_1, \ldots, f_k$ span the different frequencies. In other words, the input is a series of vectors $x(t)$ that specifies the intensity of various frequencies as time progresses. We use a many-to-one RNN, which has the general architecture shown in Figure 2.

Figure 2: Many-to-One LSTM [13]



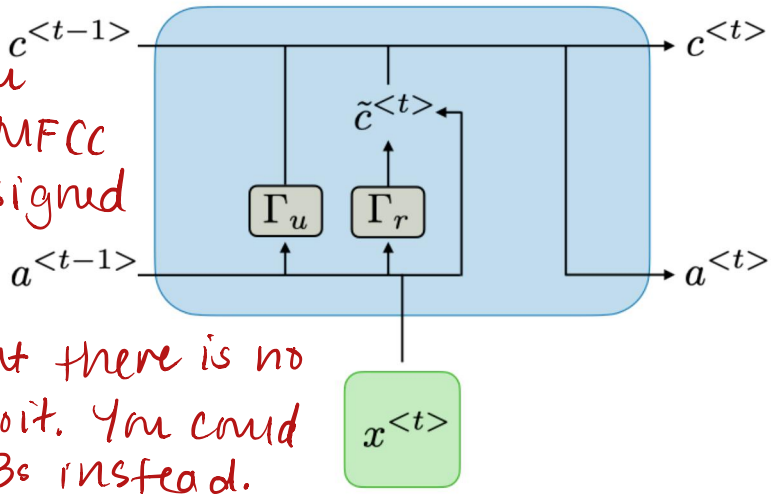
Note that the output y is a vector corresponding to the multiple accents and that, after applying softmax, we interpret y as the confidence the network has for any particular accent. The predicted accent is then $\arg\max y_i$.

A bidirectional RNN adds a second RNN going in the other direction with the time series signal reversed. This makes sense in the context of human speech since future information often changes the way we interpret prior information. The result is a backward output $\hat{y}$, which we combine with the forward output y by simply taking the softmax of $\delta y + (1 - \delta)\hat{y}$ where $\delta \in [0, 1]$ is a hyperparameter controlling how much emphasis we place on each direction.

The exact variant of LSTM that I use for the BiLSTM is not set in stone. However, I'm inclined to suggest a Gated Recurrent Unit, as shown in Figure 3. It has the advantage of being a simpler model and quicker to train than vanilla LSTMs [14].

Are you thinking of convolving only over the MFCC vector for a given time step? If so, the MFCC features are designed to be uncorrelated. That means that there is no structure to exploit. You could do this over MFBs instead. Alternatively, people generally stack MFCC/MFB frames and convolve over time instead.

Figure 3: Gated Recurrent Unit [13]



3.3 CNN + BiLSTM

We do not modify the BiLSTM model when prepending the CNN. Recall that the MFCC spectrogram M can simply be viewed as a 1-channel image, and so computer vision techniques such as the CNN can be applied. I will not discuss in much detail the exact model I plan to use for the CNN since it will almost certainly change. However, one fact should be noted.

People handle this by doing max pooling over time.

We wish to keep the BiLSTM model constant. If the original spectrogram is $m \times n$, we may end up with a $c \times k \times \ell$ output after convolution layers by using c filters. But BiLSTM accepts only rectangular arrays as input. Thus, the final step in the CNN should be taking a linear combination along the first axis to flatten back into 2 dimensions. Mathematically, let the output of the final convolution be a sequence of $k \times \ell$ matrices $Q_1, \ldots, Q_c$. Then the input to BiLSTM is given by

(this is mean pooling if $b = \frac{1}{c}$)

$$Q := \sum_{i=1}^{c} b_i Q_i \qquad \text{such that } b_i \geq 0 \text{ and } \sum_{i=1}^{c} b_i = 1 \tag{1}$$

Try both!

The values b_i of the combination are learnable parameters.

The advantage of a CNN is that it captures local information in the sense that the output of any convolution depends only on inputs that are close together in frequency and time. The output of the CNN then transfers this condensed representation of the data to the RNN for classification.

3.4 Data Augmentation

You may want to consider additional augmentation as well.

Because of the complexity of the CNN + BiLSTM model, it may be fruitful to augment the training set by adding Gaussian noise to the audio samples. It would not change the perceived accent of the individual. However, it may help capture differences in the way people speak. Even perhaps minute deviations within the same person from day to day.

3.5 Accent Classification

how big is this space?

Let $\mathcal{W}$ be the space of word utterances and Λ the space of accent labels. The model specified in previous parts of Section 3 yields a mapping $\alpha\colon \mathcal{W} \to \Lambda$. However, the input is generally a string of n word utterances, so we need to construct a map $\beta\colon \mathcal{W}^\star \to \Lambda$ using α where $\mathcal{W}^\star$ is the space of finite strings of word utterances. The most straightforward scheme is a vote where sentence $(w_1, \ldots, w_n) \in \mathcal{W}^\star$ is assigned the most frequent accent in $\alpha(w_1), \ldots, \alpha(w_n)$ with ties broken arbitrarily.

An alternative is to retain the natural probability distribution over Λ granted by the model's output. That is, we have n probability distributions $p_1, \ldots, p_n$ over Λ. We then choose the accent that received the most probability mass $\arg\max_{\lambda \in \Lambda} \sum_{i=1}^{n} p_i(\lambda)$.

Finally, we can leverage the fact that we already know the words that the recorded individual is saying. We then attempt to maximize

$$P(\beta(w_1, \ldots, w_n) = \lambda \,|\, \alpha(w_i) = \lambda_i, 1 \leq i \leq n) \tag{2}$$

The intuition being that the label of certain words may be more indicative of an accent than others. For example, words with short i and o phonetics are more likely to reveal a French accent, while words with a hard t are more likely to reveal an English accent. Of course, this relies on the fact that we already know the transcript.

I had been assuming that you had word-level boundaries. The CNN would operate at the word-level (or maybe phone-level) and then the output of this model would be fed into an RNN. You could even use attention to understand where, within the sentence, the model relied upon for the final accent prediction (e.g., for French - short i and o).

4 Ethical Considerations

4.1 A Disclaimer on Results

For the most part, an accent classifier does not make use of confidential or highly personal data, so no special consideration is needed for data collection. Of course, any system that classifies humans requires the disclaimer that it is not perfect. Even if my project hits a fantastically high test accuracy, we should exercise caution and prudence before making claims about individuals based on accent classification.

We must keep in mind that the ability to automatically tag someone with a particular ethnicity or background, in the wrong hands, can aggravate systemic discrimination. For example, there's the idiotic claim by an Israeli startup that it can use facial analysis to "know whether an anonymous individual is a potential terrorist, an aggressive person, or a potential criminal" [15]. What could *possibly* go right with adopting such a system?

4.2 Diversity and Bias

My project focuses on classifying among several accents that are frequent in available datasets. As such, it's not applicable in a diverse, heterogeneous population. It is designed to classify accent *given* that the accent falls into one of several categories. If the accent lies outsides of those categories, the system will blindly provide its best guess, but will inevitably fail. It does not make sense to use a Mandarin-Spanish-Russian-Nigerian classifier on an Indian accent. You will, with 100% certainty, get an incorrect classification.

There exists a mindboggling diversity in spoken English. Within broad accent categories such as "American", there are many variations. Texans and New Yorkers sound different. Even residents of Brooklyn and Queens speak in slightly different ways. The datasets we use are not representative of even a fraction of the many regional dialects within broad accent categories.

Last but finally not least, gender bias often pops up in human-centered AI. More often than not, this has a negative effect on women and non-binary individuals. The datasets used in this project

offer both male and female voice recordings. So while the issue still lingers, I hope that it's at least mitigated. Along the way, certain steps may need to be taken to overcome the effect that gender has on accent classification.

5 Way Forward and Schedule

5.1 Timeline

We present the following order of tasks in completing this project. Unless otherwise specified, each task requires the completion of all previous tasks. Estimated time required for each step are provided in brackets.

1. Obtain and organize the dataset. [1 week] Due to possible delays as described in Section 7.1, I'll concurrently work on the preprocessing step since it does not require the entire dataset. Rather, I only need to have a small subset to ensure that it works correctly.

2. Set up data preprocessing pipeline as in Section 3.1. [1 week] This means a data loader and transformer that loads the audio, converts stereo to mono, splits on word, augments with Gaussian noise as in Section 3.4, and applies MFCC transformation. The potential obstacle in this step is splitting by word, which likely means detecting silence.

3. Implement a GRU for one direction of the BiLSTM as in Section 3.2. [2 weeks] The other direction is the same architecture, just with a reversed signal. This can be done at the same time as the next step.

4. Convert BiLSTM output into accent classification with one of the schemes mentioned in Section 3.5. [1.5 weeks] This step is needed to actually predicts accents. It's possible that I start with a simpler classification scheme, before trying a more complex one.

5. Prepend a CNN onto the BiLSTM as in Section 3.3. [1.5 weeks] Since I've had some experience with CNN's, I expect this step to go a bit quicker. However, there's still the challenge of making sure it attaches to the BiLSTM.

6. More training, make tweaks, finalize hyperparameters, polish code, record metrics, create

charts, and prepare for presentation. [2 weeks] This step is pretty self explanatory.

All in all, the project will take an estimated $1 + 1 + 2 + 1.5 + 1.5 + 2 = 9$ weeks. This leaves some wiggle room for unforeseen issues and reading literature.

5.2 Milestone Check-In

I couldn't find exactly when the milestone check-in takes place. Assuming it happens halfway through the project at 4.5 weeks, then I should have the BiLSTM model finished or mostly finished. It should yield softmax output when provided MFCC spectrograms. It's possible I will have a simple voting scheme over the words so that I can demonstrate something more tangible for the check-in. I don't expect that the model will have stellar performance by the halfway point.

6 Final Presentation

I prefer presenting with a deck of slides combined with a (possibly prerecorded) demo. So the usual Zoom screen sharing method should suffice. The content of the presentation will roughly follow the ordering of this proposal.

1. I will outline the problem motivation, background information, and related work. Then, I will give a brief overview of my solution(s) and the dataset(s) that I used for training/testing.

2. The meat of the presentation will be a comprehensive but succinct description of the various components of my system: preprocessing, BiLSTM, CNN, classification, etc.

3. I'll follow with a demonstration of the system in action. Very likely, I'll have some form of logging so that the audience can see the result of each step.

4. The presentation will conclude with ethical considerations if an accent classification system were put into practice. I will also discuss some possible avenues for extensions and give acknowledgements where due.

I'll also maintain a Github repo with source code, documentation, slides, and a paper. If something fruitful comes out of this endeavor, I'd love to shoot for publication!

7 Tools and Materials

7.1 Dataset

There is also a large-scale corpus available through Mozilla.

The data used for this project will (tentatively) originate from the Speech Accent Archive [9] and the Wildcat Corpus of Native- and Foreign-Accented English [10]. The link to retrieve [10] is down, so it's unclear how I'm going to retrieve that data. Perhaps I could contact the Wildcat project members and see if they'd be willing to send me it. Retrieving [9] is also not entirely straightforward. Rather than a central repository that can be cloned, it'll require some web scraping. Fortunately, there seems to be an existing script [11] that I may be able to leverage.

7.2 Libraries and Frameworks

Web scraping for data would require BeautifulSoup and requests. General audio manipulation, splitting based on words, and channel averaging will use Pydub. The MFCC spectrogram can be generated with librosa. I'll almost certainly make use of the usual Python data science stack, i.e. numpy, scipy, pandas, matplotlib, etc. At the moment, I'm not sure whether to use PyTorch or TensorFlow for the actual model. I have more experience with PyTorch, but I'd like to learn TensorFlow, and this is a nice opportunity. The final accent classification, as specified in Section 3.5, may be either vanilla Python or scikit-learn, depending on implementation.

8 The Team

Since I am the only member of this team, I will perform all tasks.

9 References

[1] `https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers`

[2] Chen P., Lee J., Neidert J., Foreign Accent Classification. `http://cs229.stanford.edu/proj2011/ChenLeeNeidert-ForeignAccentClassification.pdf`