Unsupervised Phoneme Recognition

Kyle Roth   Seong-Eun Cho

December 14, 2018

1 Introduction

Speech recognition is a fundamental discipline of computational linguistics. It has been a challenge sought by many from the birth of computational devices, and has recently seen great success through the use of neural networks. Similar to speech recognition, phoneme recognition is the challenge of transcribing audio inputs into International Phonetic Alphabet (IPA). In this project, we will attempt to build an automatic phoneme recognition model. We test various statistical methods, such as principal component analysis and embedding spaces.

1.1 Related Work

IPA transcription of spoken language is a difficult process. Field linguists spend their lives training their ear to catch small differences in sound, recording them in IPA. Recorded audio has helped little. Automatic phonetic transcription has been attempted by [1], [2], and many others, but was limited to one language and didn't use neural networks. [3] used temporal flow networks to achieve 80% accuracy, but on a dataset of only 37 words spoken over 2.5 hours. Their approach used the output of an array of networks to then predict phonemes; each network was trained to estimate a specifically chosen phonetic feature.

1.2 Our Approach

We feel that these approaches were limited by the lack of data and their specificity to one language. We will build a model that is not limited by language using semi-supervised methods, allowing us to learn on varied, multilingual, unlabeled data. We will use a machine learning model to recognize phoneme boundaries on unlabeled data, using a loss function based on some phonetically transcribed data where the phonetic tags are known. We will use a temporal flow neural network as the algorithm, with distance metric learning and nearest neighbor methods to build the loss function. We will be using raw waveform data from two distinct sources:

- General Conference talks from the Church of Jesus Christ of Latter-Day Saints, and
- VoxCeleb, a corpus of interviews with celebrities from around the world.

1.3 Motivation

There are many properties of speech that cannot be fully represented in plain language text. A purely phonetic recognition model could potentially enable us to extract other information such as the speaker's dialect, small intricacies in the tone of voice, sarcasm, accenting, and other features. Also, most state-of-the-art ASR approaches use language-specific models trained on labeled data, but a model that can learn without labeled data could be a truly universal ASR model that is not limited to the language spoken.

2 Data collection

2.1 General Conference data download and conversion

The General Conference data is scraped directly from the website of the Church, lds.org. The advantage of this dataset is that a vast amount of data is directly available, and because most talks are translated into over 60 languages, we have very good coverage of languages. One disadvantage of this dataset are that it might not have a very good coverage of speakers, since the majority of the speakers in General Conference are older Caucasian males. This issue is partially resolved by the fact that translators of different languages vary in age, race, and gender. We have gathered close to 2000 hours of audio data from General Conference.

2.2 VoxCeleb data download and conversion

The bash commands in `voxceleb.sh` download VoxCeleb1 and VoxCeleb2 and perform conversion to .wav on VoxCeleb2 (natively .m4a).

```
./voxceleb.sh
```

On an 8th generation i5 processor the `ffmpeg` conversion took about 2 days.

The following function removes .m4a files and names the .wav files correctly after being converted by `ffmpeg` in `voxceleb.sh`.

```
convert.cleanup('/media/kylrth/KYLEBAK/data_project/data/vox2_dev')
```

This data was downloaded from http://www.robots.ox.ac.uk/~vgg/data/voxceleb/, and was formatted in the following way:

```
audio.get_properties('sample/00001.wav')
```

```
Input File     : 'sample/00001.wav'
Channels       : 1
Sample Rate    : 16000
Precision      : 16-bit
Duration       : 00:00:09.24 = 147841 samples ~ 693.005 CDDA sectors
File Size      : 296k
Bit Rate       : 256k
Sample Encoding: 16-bit Signed Integer PCM
```

3 Data preparation

We first normalize the sample rate of the audio to 16 kHz and convert it to .wav format, which contains the raw waveform information that can be easily imported using SciPy. This can be done using a simple, one-line `sox` command:

```
sox input/path/of/audio.mp3 -r 16000 /output/path/of/audio.wav
```

We then perform the following preprocessing steps to the data:

- Split the audio into small segments
- Fragment trimming

3.1 Splitting into segments

We believe shorter segments of around 5 seconds will be optimal for training. We split the audio samples by setting an epsilon boundary near 0. If, in a small window of audio, the maximum amplitude of the window is less than the epsilon boundary, it recognizes that segment as a pause. This window slides across the entire audio file and splits the audio into small chunks with a minimum length at the pauses. We found that the average absolute value of the audio sample times some constant to be a good boundary.

3.2 Fragment trimming

To avoid training on partial phonemes, we trim all the bits of speech that occur separate from the main body of speech, on either end of the file. This is done using similar pause recognition techniques from the previous section.

4 Feature engineering

Raw waveform is not a very good representation of the characteristics of sound. Since our objective is to take in an audio sample and output the phonetic transcriptions, we need a way to embed each phoneme as an element of a phoneme space. We theorize that the following methods can help us to extract these features from the raw waveform:

- Discrete Fourier Transform (DFT)
- Spectral analysis
- Cepstral analysis
- Wavelets

Each phoneme has unique frequency properties, so taking the Fourier Transform of an audio sample (mapping it from a time domain to a frequency domain) helps to extract these features. The spectrum, which is defined as the log of the absolute value of the DFT, can further help extract the features as it scales the amplitude information in a logarithmic scale.

We have also tried taking the power cepstrum, which is the inverse DFT of the spectrum. Cepstral analysis is widely used in signal processing, especially in the analysis of human speech. However, we weren't able to get good results from the cepstrum, so results are not included here.

4.1 Embedding spaces

For a set Ω of objects (e.g. audio, images, or text), an **embedding space** (V, F) is defined by a metric space V and an invertible function $F : \Omega \to V$ that maps objects from Ω to the vector space V. The goal is to "embed" these objects in an n-dimensional space where metric distance and vector addition become meaningful in some way.

With phonemes, the ideal would for distance to correlate with phonetic differences between two phonemes. In addition, meaningful vector addition would imply the following illustrative example:

$F([\mathrm{v}]) - F([\mathrm{f}]) + F([\mathrm{p}]) \approx F([\mathrm{b}])$.

(The difference between [v] and [f] is voicing, and when voicing is added to [p] the result is [b].)

Finding such a function F is extremely valuable in creating a loss function for unsupervised phonetic transcription, as will be explained later on.

4.1.1 The optimization problem

The following is a summary explanation of the concepts found in [3].

If each object can be represented as a vector $\mathbf{x} \in \mathbb{R}^n$, we can formulate finding the embedding function as an optimization problem over the space M_n of matrices of size $n \times n$. If $x, y \in \mathbb{R}^n$ and $A \in M_n$ such that $A \succeq 0$, define the following distance metric:

$$d_A(x,y) = ||x-y||_A = \sqrt{(x-y)^\top A(x-y)}.$$

In addition, let S be a set of pairs such that for each pair $(x_i, x_j) \in S$ it is known that x_i is similar to x_j. Similarly, let D be a set of pairs such that for each pair $(x_i, x_j) \in D$ it is known that x_i is dissimilar to x_j. Then the optimal embedding function is $A^{1/2}$, where A is the solution to the following optimization problem:

$$\begin{array}{c} \max_A \sum_{(x_i,x_j)\in D} ||x_i - x_j||_A^2, \\ \text{subject to } \sum_{(x_i,x_j)\in S} ||x_i - x_j||_A \le 1, \\ A \succeq 0. \end{array}$$

We can use gradient descent (or stochastic gradient descent if S and D are large) to optimize A. The algorithm is described in detail in [4].

Here is an example of how predicting the phoneme in a sound should work:

```python
# if `phonemes` is a dictionary mapping phoneme characters to lists of SoundWave objects,
# the following will work.
rate, wave = wavfile.read("ipa/16k-wav/pulmonic_consonant/Voiced_bilabial_plosive.wav")
sample = audio.SoundWave(rate, wave[3079:4491])
recognizer = PhonemeSimilarity(phonemes)
print(recognizer.predict(sample))  # should print 'b'
```

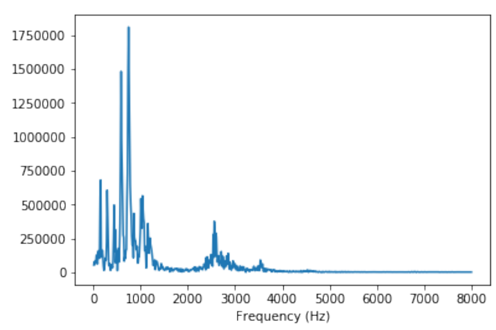
The algorithm in [4] uses gradient descent with iterative projections to meet the constraints, but we were unable to implement the similarity constraint by the time of writing. The matrix equation representing the problem appears to be dense, meaning that each projection would require inverting an $n^2 \times n^2$ matrix, which is prohibitively difficult. We also attempted to use gradient ascent on the difference between the dissimilarity term and the similarity term, to no avail. Further work will continue in January.

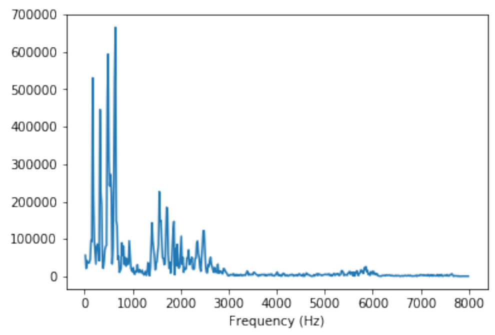
5 Data Visualization

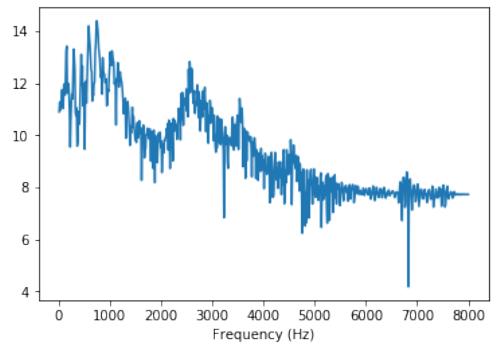
5.1 Discrete Fourier Transform and Spectral Analysis

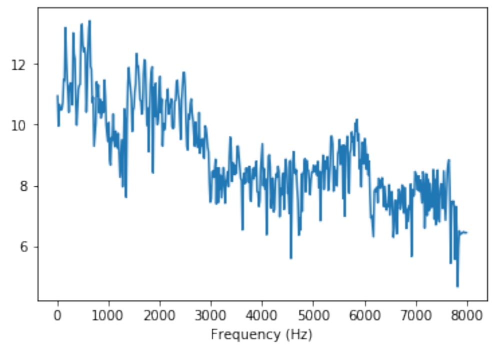
From Wikipedia, we have gathered audio files of a person pronouncing each phoneme. Here, we pinpoint the exact timestamps at which the phoneme sound occurs from the phoneme audio and compare it to the occurrences of the respective phoneme from the data. We do this for the phonemes [b], [f], [n], [æ], and [i]. We take the DFT and the spectrum of the phonemes, and compare the results side by side. Doing this reveals that each of the phonemes show similar characteristics in the DFT space and spectrum space even when spoken by different speakers.

Below are the DFT (left) and the spectrum (right) of two samples of [B].

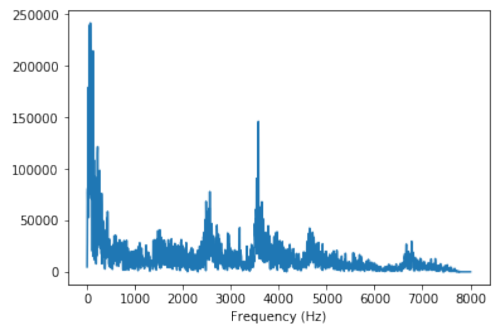


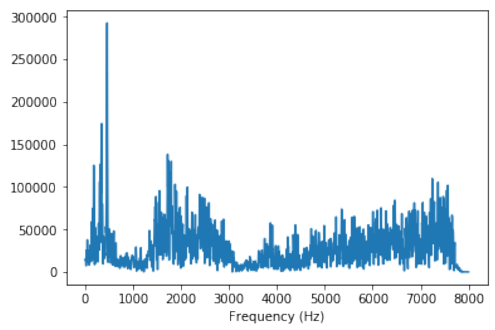


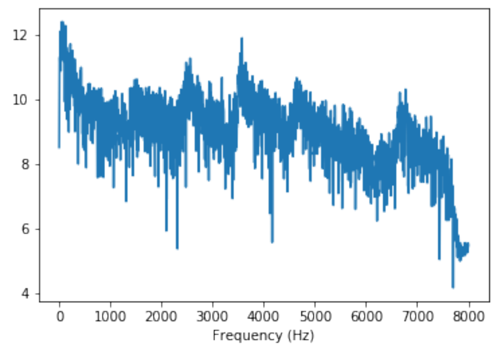


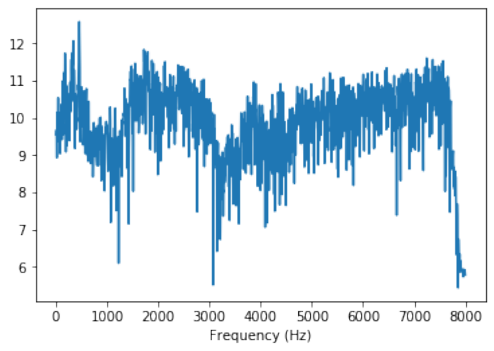


[f]:

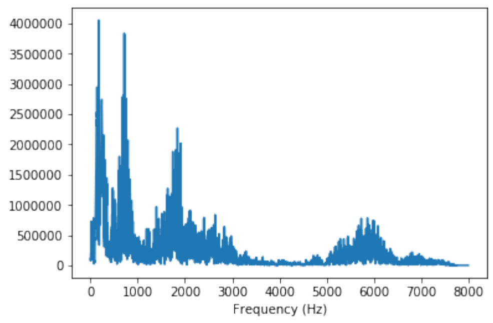


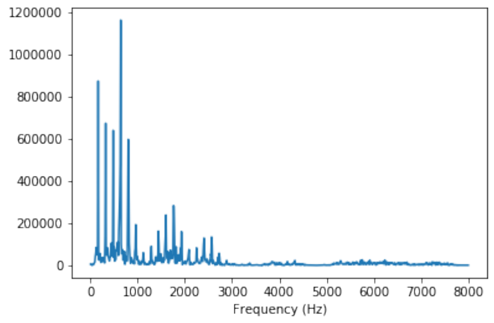


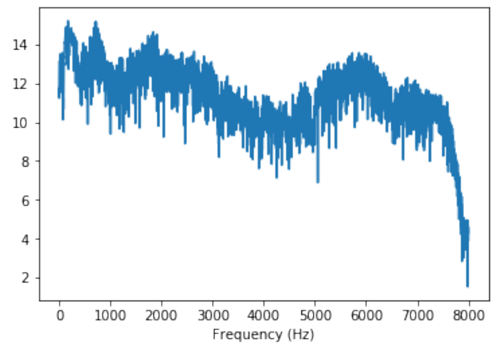


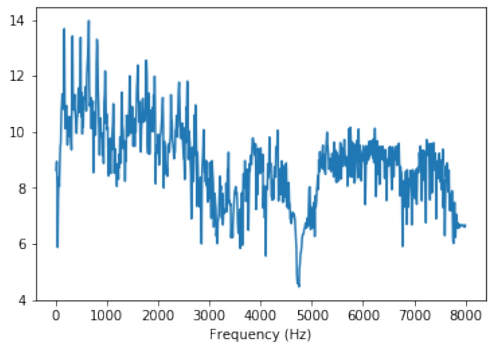


[æ]:



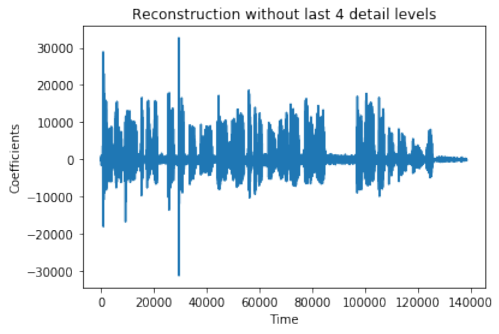






5.2 Wavelets

Speech recognition using wavelets has been done previously, with moderate success [5]. Here we use the Discrete Wavelet Transform to represent the audio with less detail than before.

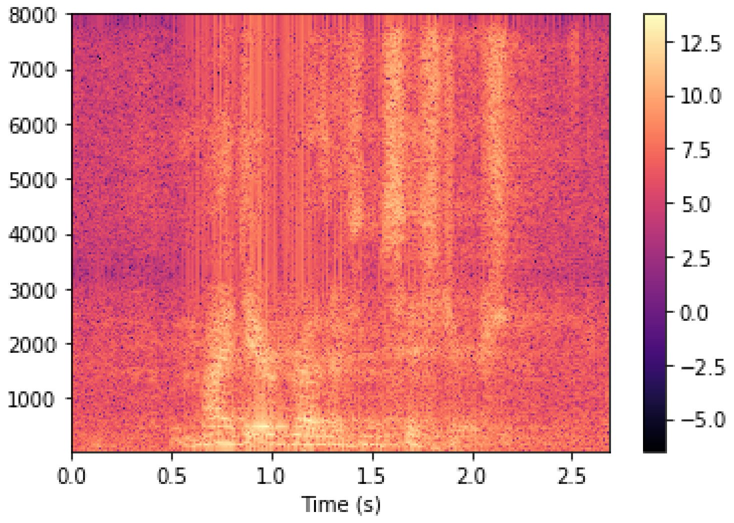


5.3 Spectrogram

Spectrograms are incredibly useful for visualizing audio recordings of human speech because they capture the nature of phoneme frequencies as a visual graph. Spectrograms are computed in the following way:

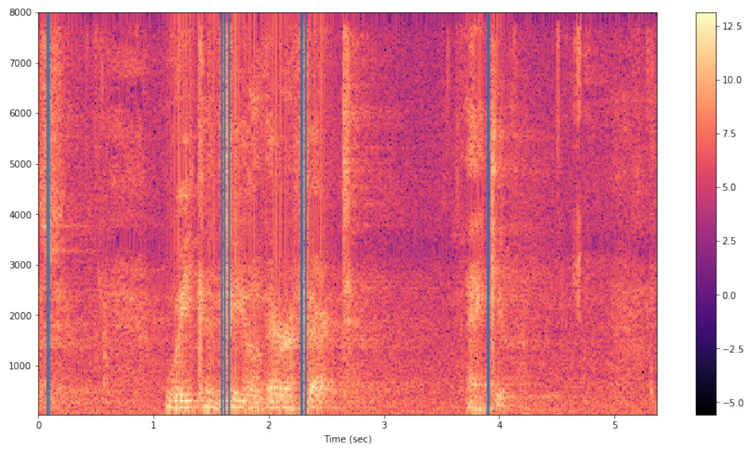
A small window of a specified time frame (defaulted to 0.03 seconds) is initialized at the beginning of the audio sample. The spectrum $\log(|\mathrm{DFT}(x)|)$ of the audio in that window is computed and saved. This window slides across the audio sample with a small amount of overlap from the previous window (usually to 0.01 seconds). This continues until the entire sample has been covered, and is returned.

We transpose the returned spectrogram before plotting, since each element of the array corresponds to the spectrum of a window. We plot the spectrogram over time and frequency using a heat map.



5.3.1 Phoneme detection using normed differences

As we have seen from the spectral analysis, one naïve way of embedding phonemes is by taking the spectrum of it. We then hypothesized that if each phoneme has similar values in the spectrum space, then by taking the model phoneme and taking the L2 norm difference with each window of sample of our data will reveal that the places where the phoneme occurs in our sample will have a very small norm difference. We used this idea to identify 10 smallest norm difference between a window of sample and the model phoneme for [f].

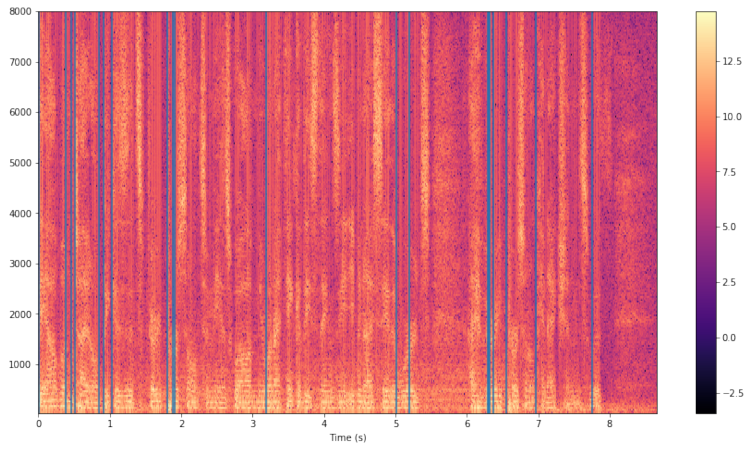


From the result above, we see that the 10 lowest norm difference occurs in 4 distinct places. In this audio sample, the speaker says “We can feel their effect, thank you.”. The first occurrence near 0 is a trailing [s] sound from the previous audio which is a fricative like the [f]. The second and third occurrences at around 1.5 and 2.2 seconds respectively are precisely when the speaker says [f] in the words “feel” and “effect”. The last occurrence near 4 seconds seems to be an outlier. From these results, we can see that even with a naïve approach, we can identify the characteristics for phonemes.

5.3.2 Phoneme detection with derivatives

We want to find a sequence of time stamps which divide the audio into phonemes. We’d like to test the hypothesis that phoneme boundaries occur where the change in certain features is most acute; that is, where the derivative along some dimension of the feature space has the greatest magnitude.

We don’t have the expertise to mark phoneme boundaries by hand, and we feel that even expert human judgment is prone to produce errors. Thus we judge the output of various attempts simply by checking whether the division is reasonable to the human ear.

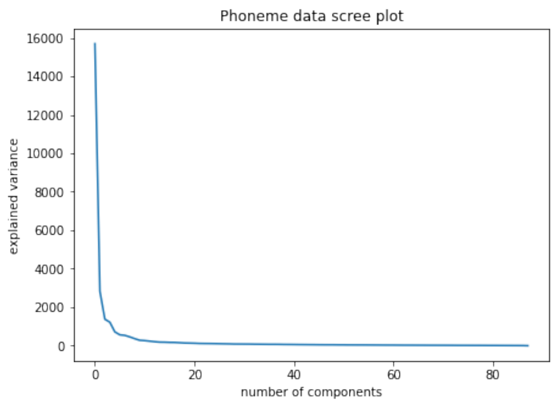


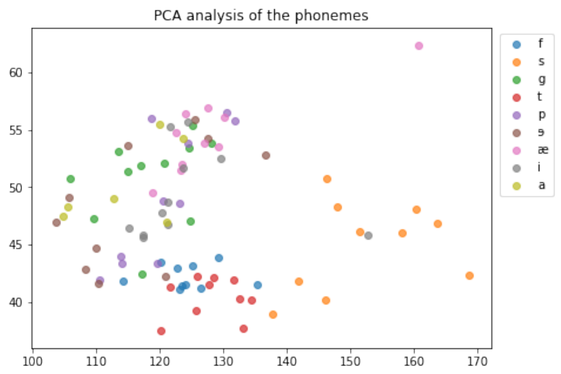
Upon inspection, there are several points that seem to be consistent with boundaries between morphemes, but more analysis needs to be performed. There seem to be many false positives.

5.4 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical procedure that allows least-squares projection of the data points into a lower-dimensional subspace that preserves most of the cumulative variance. Specifically, if X is an $n \times d$ matrix of n data points with d features, this is done by finding the singular values of the square matrix $X^\top X$ and choosing an $s < d$ where the first s singular values contain the most of the cumulative variance. We use the first s components of the orthogonal matrix V from the SVD to project the d-dimensional data points into a much lower s-dimensional projection. It is useful in compression and noise reduction, but also very useful for data visualization as we can take a high dimensional data points and project them onto a two dimensional graph.

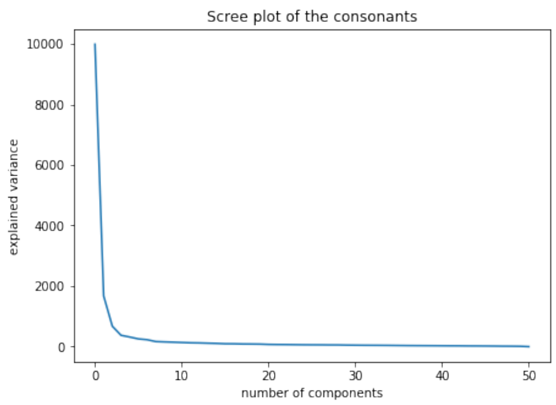
In order to perform PCA, we manually picked out around 10 instances of the following phonemes [f], [g], [s], [p], [t], [ə], [i], [æ], [a] spoken from our data. We picked out 88 samples of the above phonemes. We computed the spectrum of each, then resampled them to 240 samples to create identical dimensionality across all of the phoneme data. We found that increasing the number of samples did not affect the results of PCA significantly. We first computed for the explained variance of each component and produced a scree plot.

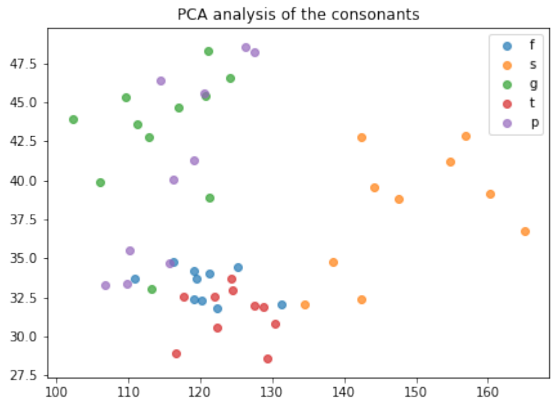




Observe that even though the dimension of the features is 240, we can only find 88 components because we only had 88 phoneme data which is less than the feature dimension. We found that 63.6% of the cumulative variance is contained in the first two principal components, and in order to contain at least 90% of the variance, we would need 23 components. We then computed for the first two principal components and plot the result as a scatter plot.

Although only 63.6% of the variance is contained in the first two components, we can still observe a general grouping among phonemes. There seem to be a good separation among the phonemes [s], [t], and [f], while the others had less separation. This motivated us to perform PCA on the consonants and the vowels separately.

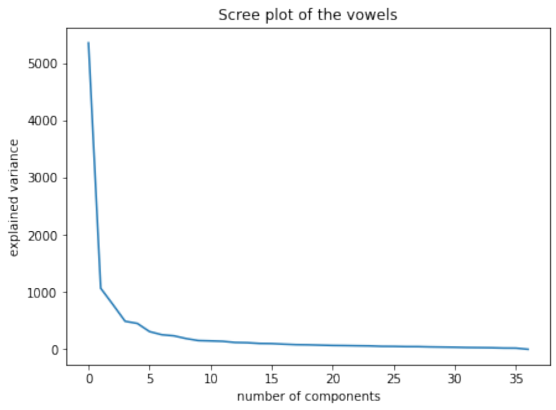


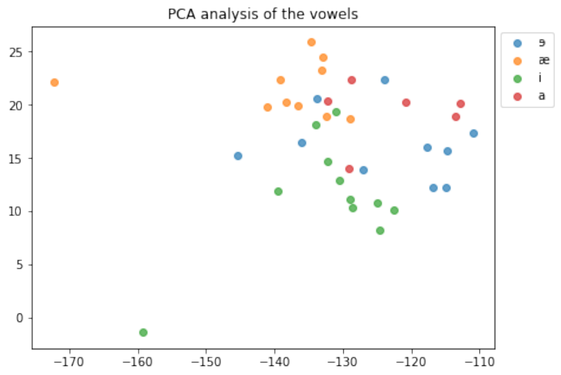


We found that when only looking at the consonants, 71.8% of the variance is contained in the first two components. We performed 2-variable PCA in a similar manner.

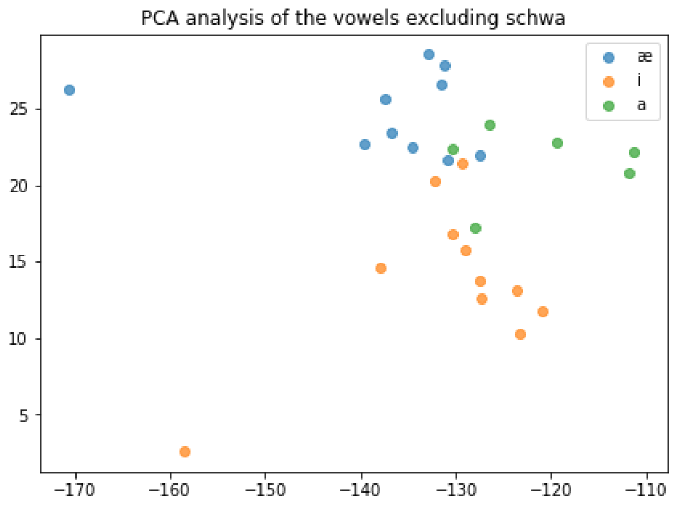
Analyzing just the consonants, there seems to be a pretty good separation among each phoneme. Specifically, we find that fricatives (consonants produced by forcing air through a narrow channel of the mouth), which are [f] and [s], are relatively close to each other while the plosives (consonant in which the vocal tract is blocked so that all airflow ceases), which are [g] and [p], are also close to each other. Interestingly, [t], which is a plosive consonant, lied closer to the fricatives. This can be partially explained by the fact that the American accent of [t] is very aspirated (meaning strong burst of breath accompanies the consonant), which mimics some properties of a fricative. The fact that the [s] and [t] are so close to each other could also be explained by the fact that they are both alveolar consonants – consonants articulated with the tongue against or close to the superior alveolar ridge.

We then perform a similar analysis on the vowels.





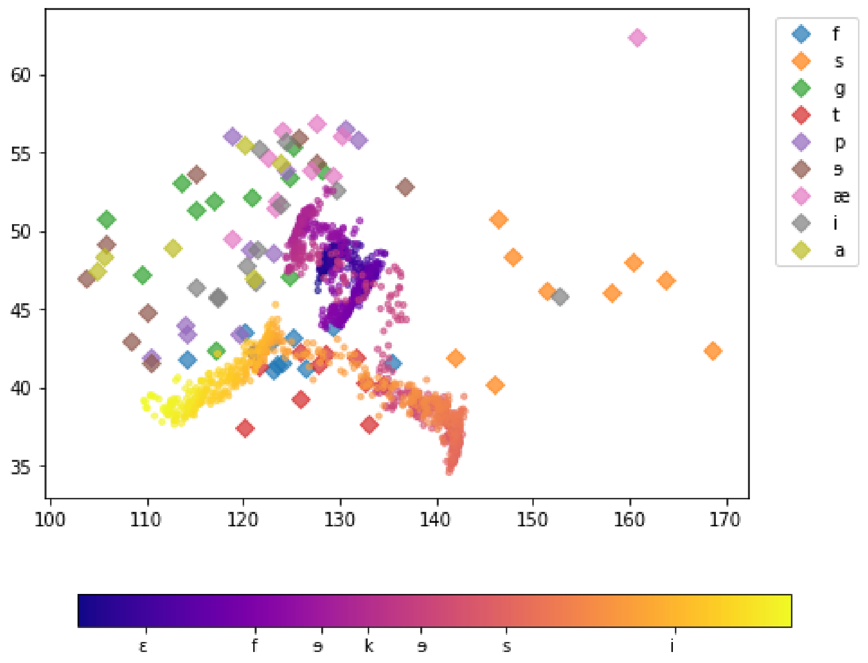
For the vowels, only 58.3% of the variance is contained in the first two components. Surprisingly, we can still see that same phonemes are close to each other while being somewhat separated from different phonemes. The one phoneme that seems to be more scattered through out is [ə] (called "schwa", this sound is made when pronouncing "o" in "personal"). This is rather expected, since [ə] is the most neutral vowel. We wondered if excluding [ə] produces better results for the vowel.



As expected, without the [ə], the boundary between different vowel phonemes seems to be clear. In phonetics, vowels are classified by the place of the mouth (front to back) and the openness of the mouth (open to closed) when the sound is made. With more data, these properties might become evident even in the 2-variable PCA.

5.4.1 2-variable PCA interpolation

We took a same clip of some audio data to see if we could perform an interpolation in the 2-variable principal component subspace. The sample audio is of a speaker saying the word “efficacy”, which contains the phonemes [ɛ], [f], [ə], [k], [ə], [s], and [i].



In the diamond scatter plot are the same points from the 2-variable PCA of the entire phonemes. The smaller points represent the value of the spectrum in the 2-variable subspace as the speaker pronounces the word. The continuous color map shows progress through time. We can see that the first phoneme [ɛ], which is a vowel starts closer to where other vowel phonemes are, then as the phoneme changes to [f] it is approaches the other [f] points in the space. It then moves back towards the area of vowels and [g] as the phonemes [ə] and [k] are pronounced, then quickly moves over to the area around the [s]'s as [s] is pronounced in the word. Finally, it drifts away back towards other vowels as [i] is pronounced.

6 Conclusion

We have shown evidence that the spectrum of the audio could be a suitable element of the preprocessing step for phoneme embedding. Further analysis through PCA shows that certain projections of the spectral features are semantically significant. In the future, we will improve upon this work using the distance metric learning algorithm, which could potentially allow us to perform an unsupervised recognition of phonemes.

References

[1] Van Bael, et al., *Automatic phonetic transcription of large speech corpora*. Computer Speech Language, Volume 21, Issue 4. 2007 Oct. Pg 652-668.

[2] Schiel, Florian. *Automatic phonetic transcription of non-prompted speech*. Department of Phonetics, University of Munich, Germany. `epub.ub.uni-muenchen.de/13682/1/schiel_13682.pdf`.

[3] Chang, et al. *Automatic phonetic transcription of spontaneous speech (American English)*. International Computer Science Institute, UC Berkeley. ICSLP 2000, Beijing, China.

[4] Xing, et al. *Distance metric learning, with application to clustering with side-information*. UC Berkeley, 2003.

[5] Unknown authors. *Speech Recognition using Daubechies Wavelets* `clear.rice.edu/elec431/projects97/Dynamic/main.html`