Speech Recognition of Common Words

Marina Knittel

Friday, May 12th, 2017

1 Introduction

Speech recognition is used in many different aspects of our daily lives, from voice identification systems to Apple's Siri. Due to the common use of speech recognition technologies, programmers and technology users alike should try to understand speech recognition, how it works, and when it does not work as well.

The purpose of this project is to explore the basic underlying factors and processes in common speech recognition systems. This paper will describe a program that constructs and evaluates a simple Hidden Markov Model with Gaussian Mixture emissions in order to model small sets of common spoken words. As an input, the program takes raw audio files labeled with the beginning and ending timestamps of every phoneme and word within, as well as a transcript of the spoken sentences. The programs then extracts 13 Mel-frequency cepstral coefficients (MFCCs) and creates the aforementioned model to distinguish between words it was modeled after and other words. We discuss four different tasks used to evaluate the models' varying effectivnesses in different situations, and also present findings on the effect of adjusting the number of Gaussian Mixture emissions used in the model on its performance.

2 Methods

The speech recognition process in this project was primarily based on the language processing systems described by Jurafsky & Martin (2009), with some additional direction from Huang et al (2001). The models were trained and evaluated on data

from TIMIT titled "TIMIT Acoustic-Phonetic Continuous Speech Corpus" by Garofolo et al released in 1993. This data set consists of 630 speakers of 10 basic English dialects uttering 10 phonemically varying sentences (although no two speakers had the same set of 10 sentences). The data set included audio files of each sentence, a transcript of each sentence, files marking the beginning and end of each word and phoneme in the audio file, and some basic information about the speakers themselves.

The data extraction process varied by task. I created four different tasks to evaluate the effectivenesses of the resulting models in different circumstances. All tasks had the goal of identifying words by extracting the section of the audio file corresponding to those words and trying to model those words. It would then compare the likelihood that the task words or two "gibberish" words (suggested by Peter Mawhorter) were generated by the model. The gibberish words were two frequently used words that the model wasn't trained on for the given task.

The first task used the top four most frequent words, where frequency is measured by how many times the word was uttered in the unique sentences from all audio files. This included "the", "a", "to", and "of", with a total of 31071 instances between all words and 15 unique phonemes. I will refer to this as the Top Four task. The second, the Top Two task, used the top two most frequent words. This included "the" and "a", with 16163 total instances and two unique phonemes. The third, the Three Alike task, used three similar-sounding words in the list of top ten most frequent words. This included "in", "is", and "it", with 19572 instances and four unique phonemes. The final task, the Four Different task, used four words from the top ten most frequent words that had little phonemic overlap. These were "the", "to", "and", and "he", with 30359 total instances and nine unique phonemes. Different models were constructed for each of the four tasks.

My program started out by extracting the location of the audio files directory, the sentence transcript, the word and phoneme timestamps, and some basic information about the speaker. It then extracted the audio file data using a scipy library. Using the word timestamps and the frequency of the recording, it located the start and end of each word in the data set. It used the starts and ends to extract the portions of data set corresponding to the current task words. It then used the "python_speech_features" Python library to extract the MFCC features for every 25 ms window. 25 ms was the suggested window time provided by Jurafsky & Martin. MFCC features, in general, are found by taking the Fourier Transform of the data, mapping the results onto something known as the Mel scale, taking the logs of these,

performing another transform called the discrete cosine transform, and then finding the resulting amplitudes. The result yields what is known as the acoustic vector, a 13-dimensional vector known to be useful for representing phonemes in audio recordings (Muda et al, 2010). I continually added these vectors to a matrix of data and kept track of how many data points (number of windows) each word recording consisted of, and what word it was.

After this, I divided the data into training and testing such that 80% of the recordings were used for training data. This did not correspond to precisely 80% of the data points, because each recording had different numbers of data points based off of the recording length. The number of data points is how many 25 ms windows it can be divided into. The expected number of data points used for training is 80%, though.

Next, based off the advice from Jurafsky & Martin, I created and trained a Hidden Markov Model using Gaussian Mixture emissions using "hmmlearn"'s "GMMHMM" class. I used the number of components equal to the number of unique subphonemes (three times the number of unique phonemes) across the words in the given task, as suggested by Jurafsky & Martin. The Hidden Markov Model is a natural selection for speech recognition because speech consists of a time-varying sequence of individual components and subcomponents. For instance, a word can be divided into phonemes, which can be divided into beginning, middle and end components known as subphonemes. HMMs are specifically used for modeling components of sequences of data and identifying the probabilities of transitioning from one component to another in a sequence. In terms of this speech recognition project, we are modeling subphonemes and the probabilities of transitioning between subphonemes to create words. Specifically, the model should contain subphonemes as hidden states. Ideally, an HMM used for speech recognition consists of sequences of transitions between subphonemes that take place within words used in the given data set, as well as loops to iterate over the same subphoneme multiple times to take into account slower speech (Jurafsky & Martin, 2009). However, the library I used did not allow me to control which states could transition to which states. The hope is that after training the model, it would have more or less the aforementioned desired structure.

The Gaussian Mixture emissions were selected based off the advice from Jurafsky & Martin. According to them, other emissions are problematic because acoustic models are continuous and real-valued. Thus emissions like vector quantization, which are too categorical, are less ideal than Gaussian Mixture emissions, which use continuous likelihoods of observations. Thus, Gaussian Mixture emissions are more commonly

used in speech recognition, and so I decided to use this for my project (Jurafsky & Martin, 2009).

To analyze my results, I varied the number of GMMs from five to 55 by 10s to see the relationship between the performance of the model and the number of GMMs used. I fit the Hidden Markov Model to the training data, and found the log likelihood of the data sets that correspond to each word in the training and testing data sets. Then, with guidance from Prof Wu, I used a score metric that is equal to the mean of the log likelihoods of the data (corresponding to each word) scaled by $\frac{1}{n}$, where n is the number of windows in the recording. This score is important because longer sequences of data tend to have smaller likelihoods. That happens because we ask how likely it is for the HMM to produce more data points in a sequence when the sequence is longer. Since these are logs of probabilities between 0 and 1, the small likelihoods have log likelihoods that are negative with larger magnitudes. However, we do not want to compare values that depend on the length of the input. The score metric circumnavigates this by raising the likelihoods to the $\frac{1}{n}$ power (or, equivalently, by scaling the log likelihood by $\frac{1}{n}$), thereby taking the geometric mean of the data points in the given recording. Thus the likelihood we get is an averaged likelihood, and should not grow with the number of data points in the recording. Thus, if we take the arithmetic mean of this across all words, we can directly compare the scores of different models on the data without worrying about the effect of the discrepancies in recording length on scores.

In addition, I used these results to find the optimal number of GMMs, right before the models overfit the training data. This was done by graphing the test and training scores across all numbers of GMMs. If the training score continually increases, but the testing score increases, hits a maximum, then decreases, then the number of GMMs at this maximum would be considered optimal. For each task, I found this optimal number, and used it as a parameter to build and score a model for the train, test, and gibberish data. We would expect to see the training data with a better (lower magnitude) score than the testing data because the model was fit to the training data. If our model was able to accurately represent the specific words in the given task, then we would also expect the testing data to have a better score than the gibberish data. This would mean that the new instances of trained words would more likely be generated by our model than other words. The results can be seen in the next section.

3 Results

In our first analysis, we looked at the how the number of GMMs affected the scores of the training and testing data. An example of the results is shown in Figure 1, where we map the scores of the training and tesitng data for the Top Two task. As expected, we noted a continual rise in the scores of the training data across all tasks as we increased the number of GMMs. This is illustrated in the lefthand graph of Figure 1. It indicates that increasing the number of GMMs adds complexity to the model, thus allowing it to fit the training data better and better as the number of GMMs increases.

For the testing data, we consistently saw a peak in the scores at 15 GMMs. This was true for all tasks except the Top Four Task, for which there was a plateau between 15 and 25 GMMs, indicating that the peak was somewhere in that range. Similarly, the peak at 15 GMMs for the other tasks only tells us that the peak probably occurred between five and 25 GMMs, because we tested with intervals of ten GMMs. For simplicity, we will approximate all peaks as happening at 15 GMMs. We can then say this is the point at which adding GMMs makes the models fit the test data worse, and thus this is the point at which overfitting begins.

We can use these results then to say that 15 is likely the best selection for the number of GMMs parameter. We use this for the second part of our analysis: comparing the scores of the training, testing, and gibberish data for the models for each task. These results are shown in Table 1. To interpret this data, we have to consider what scale this is. The score is an arithmetic mean of the logs of the geometric means of probabilities that the models would generate single data points (representing 25 ms windows of audio). Thus, this has a logarithmic scale. So if we have scores x and y such that $x = y + \delta$, then this really indicates that the average likelihoods, p_x and p_y, had the relationship $p_x = 10^{\delta} p_y$. In addition, recall that a higher probability is more desirable (it means the model was more likely to generate the input data), and this would result in log likelihoods of smaller magnitude, which would in turn result in scores of smaller magnitude. Thus scores of smaller magnitude represent data that was better represented by the models. We can use these to interpret the scores from Table 1.

The first thing to notice is that for every task, the train data has a better score than the test data, and the test data had a better score than the gibberish data. First, this means the model fit the training data better than testing data, as ex-

pected. More specifically, the score differences between training and testing data ranged from 0.7 to 1.1. This indicates that the likelihoods of the training data were larger by a factor of 5.01 to 12.59, which is a large enough difference to say it represented the training data better than the testing data.

Second, it fit the testing data better than the gibberish data, with score differences ranging from 0.8 to 2.5. This indicates that the likelihoods of the testing data were larger by a factor of 6.31 to 316.23. These are large enough differences to say that the models represented the testing data better than the gibberish data.

Note that the factor of 316.23 came from the Top Four task (and the Top Two task was similar at 251.19). The Top Alike and Four Diff tasks had significantly lower differences between the test and gibberish scores (with factors of 12.59 and 6.31 respectively). This indicates that the models were much better able to distinguish the Top Four and Top Two task test set and gibberish set than the Top Alike and Four Diff test set and gibberish set.

If we just use this statistic to evaluate our model, we would say that it makes sense that the Top Four and Top Two tasks were better modeled by our program. This is because these data sets both had the most examples per word, and models are generally better able to represent populations when they are given larger samples. It is interesting to note that the models performed better on the Top Alike task than the Four Diff task. This may be expected because it had significantly fewer phonemes to model (four versus nine), and thus was able to distinguish task words from gibberish words partially by nature of the presence or absence of that phoneme. However, it is also noteworthy that the Four Diff task had more total data. This did not seem to have a large enough effect to make the Four Diff task easier. At this point, it is unclear why the Top Alike task was easier than the Four Diff task.

In addition to looking at the differences in train, test, and gibberish scores, we can also compare the raw scores of the test data for each task. In this case, we see a different pattern. The best scores were in the Top Four and Four Diff tasks, and the Top Alike task only had a score that differed from those by 0.2 (a factor of 1.58). The Top Two task had a somewhat worse score. The difference between the Top Two task and the Top Four and Four Diff tasks was 0.6 (a factor of 3.98). These results represent that the models for the Top Four, Top Alike, and Four Diff tasks were more likely to generate the test data than the Top Two model. However, these results are not as strong as the previously analyzed results (the difference between

the test and gibberish scores), and they do not take into account how much better the models are at representing the testing data than gibberish data. Thus, these results are not as valuable.

4 Conclusion

In this project, I tried to get a better understanding of the underlying structure of speech recognition systems, as well as what makes them perform better or worse. I did this by constructing a program, primarily based off direction from the Jurafsky & Martin book, to analyze and model spoken word data from the "TIMIT Acoustic-Phonetic Continuous Speech Corpus". To evaluate this program, I created four tasks, the Top Four, Top Two, Top Alike, and Four Diff tasks, which consisted of different sets of words from the data.

The program began by calculating the Mel-frequency cepstral coefficients of 25 ms windows of recordings of words in the given task. It then constructed multiple GMMHMMs with five to 55 GMM emissions, and calculated a scoring metric to evaluate the performance of the different GMMHMMs on the various tasks. This was done for the training and testing data. I used these results to see when the models began to overfit the training data, which was always at about 15 GMM emissions.

After this, I took the GMMHMMS with 15 GMM emissions, and evaluated how well they performed on the testing, training, and gibberish data using my scoring metric. These scores were used to evaluate how well the models were able to fit the data for the different tasks. The models were better able to distinguish the test and gibberish data for the Top Two and Top Four tasks. However, the models had worse raw scores for the Top Two tasks than all others. Because the latter statistic did not show as large of a difference between the performance for the different tasks (based off of the magnitude differences), we base our results off the former statistic, and say that our models performed better for the Top Two and Top Four tasks. This is likely due to the fact that these two tasks had larger data sets.

Our results show that the GMMHMM process with MFCC feature extraction was able to create reasonable models that could both identify specific spoken words as well as distinguish words it was trained on from words it was not trained on. In addition, based off of the differences in the performances of our models on the four tasks, we can see that the GMMHMMs performed better on larger data sets. While our speech recognition system was a much smaller scale project than most modern

speech recognition systems, we can use the results from this program to theorize that the GMMHMM process used here would likely work for larger, real-world data sets. Also, the input data size would probably be a major factor in the performance of such a model. Although we cannot be sure that such an extrapolation is reasonable, these results at least provide a basic understanding of what speech recognition process can be like and what factors may be important.

5 Acknowledgements

I would like to acknowledge Prof Wu for helping me understand Hidden Markov Models and the speech recognition process in general, as well as suggesting the scoring metric. I would also like to thank Prof Mawhorter for giving me the idea of comparing the test data to the gibberish data, and Prof Medero for providing the textbooks by Jurafsky & Martin and Huang et al.

Figures and Tables

	Score				
Task	Train	Test	Gib	Train - Test score	Test - Gib score
Top Two	-47.4	-48.1	-50.5	0.7	2.4
Top Four	-46.9	-47.5	-50.0	0.6	2.5
Top Alike	-46.6	-47.7	-48.8	1.1	1.1
Four Diff	-46.6	-47.5	-48.3	0.9	0.8

Table 1: Scores of different data on different tasks, and the score differences.

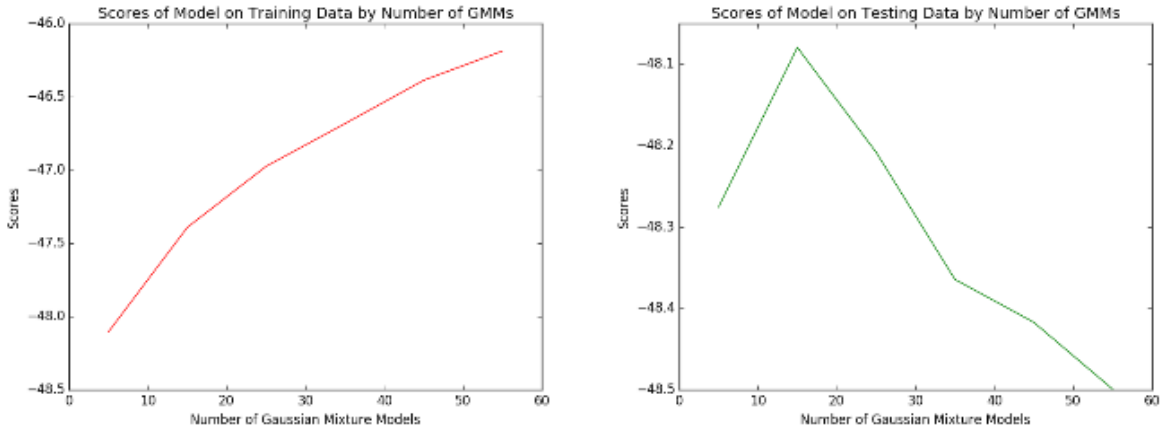

Figure 1: Graphs of the scores of the models trained for the Top Two task with five to 55 GMMs. Scores on the left are for the training data set, showing that the more GMMs, the better the fit on the training data. Scores on the right are for the testing data set, showing that at more than 15 or so GMMs, the models begin to overfit the training data.

References

[1] Dan Jurafsky, James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2009. Book.

[2] Xuedong Huang, Alejandro Acero, Hsaio-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. 2001. Book.

[3] John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, Nancy L. Dahlgren, Victor Zue. *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. 1993. Web.

[4] Lindasalwa Muda, Mumtaj Begam, I. Elamvazuthi, *Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques*. 2010. Web.