SpeakerTagger: A Speaker Tracking System

Jordan Cazamias
jaycaz@stanford.edu

Naoki Eto
naokieto@stanford.edu

Ye Yuan
yy0222@stanford.edu

1. ABSTRACT

Our project is a speaker tagging system, which can distinguish between speakers in a recording of a conversation. We applied various supervised and unsupervised machine learning algorithms such as Support Vector Machines, Neural Nets, Gaussian Mixture Models, and KMeans Clustering, to assign each segment of audio data to the correct speaker. We also tried various combinations of different acoustic features like spectral flatness, loudness, and MFCC. Overall, Neural Nets worked best on average, with roughly 56% accuracy. SVMs had the best maximum performance, with one file in particular classified at nearly 99% accuracy. However, for all approaches, there was a very wide variance in performance. Most approaches swung widely between roughly 20% and 90% accuracy for different sound files. Our next step to further develop our system would be to find the root cause as to why certain files resulted in such poor performance. After that, additional optimizations could be made to create a system that can work in real-time rather than on a pre-recorded sound file.

2. INTRODUCTION

Recording a conversation such as a business meeting, a news broadcast, or an interview can help preserve important conversational moments, but working with audio data is a frustrating task. When you actually have to go back through an archive of recordings, looking for one particular moment, it's quite the chore since audio is not easily searchable. However, if you could transcribe the speech in your audio files, suddenly navigating through them becomes a breeze. This is why so much research work is being put into systems that can automatically extract the text from audio. Beyond that, however, there are other useful pieces of metadata that would make transcripts much more rich and useful.

One important example: for a recording of a conversation with multiple speakers, it's not just useful to know what's being said, but who's saying it. This is the focus of our project.

We have created a system to perform *speaker diarization*, the process of distinguishing between the speakers in a recording of a conversation. We will also informally refer to it as *speaker tagging*. It is primarily approached as an unsupervised learning problem, although supervised approaches are possible. The main input to the system is a raw audio file of a conversation, and the output is a timeline showing when Person A is speaking, when Person B is speaking, and so on. Other metadata can be provided in the input as well: for instance, the number of speakers can be directly fed into the system to save time, although it's possible to guess this value using a parameter search. Additionally, for a supervised approach, part of the speech file is hand-annotated with speaker times so the system can attempt to fill in the rest.

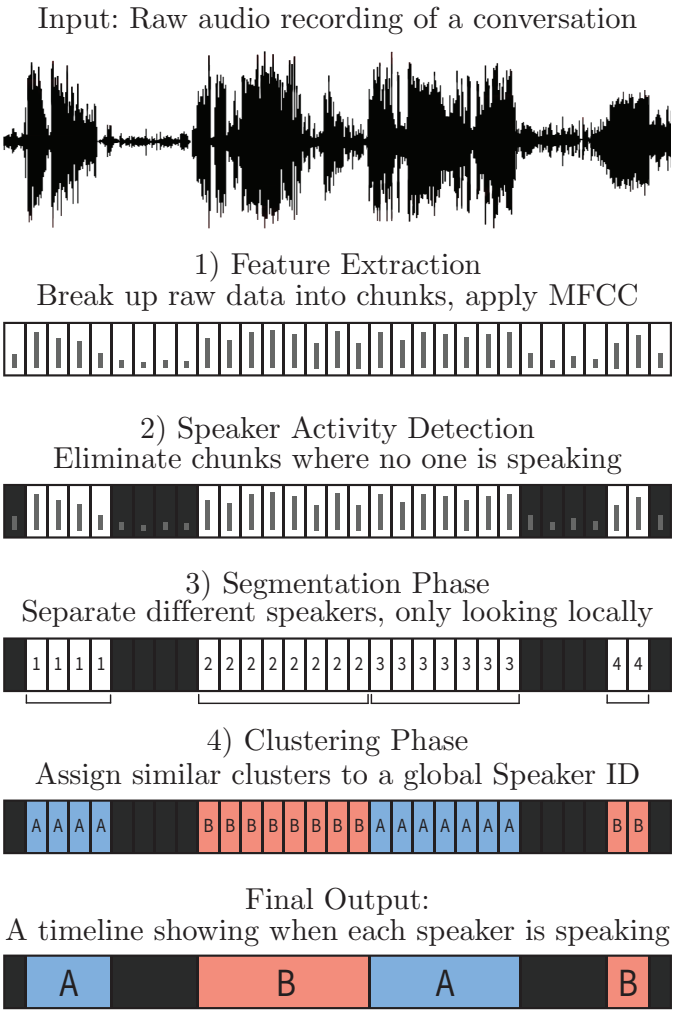

Figure 1: The general steps of a bottom-up diarization algorithm. Note that some of these steps may be ignored or merged together.

2.1 Related Work

The primary influence on our diarization system was Anguera et. al.'s literary review of the field, *Speaker Diarization: A Review of Recent Research* [1]. In it, the general diarization

algorithm is presented (as shown in figure 2) and various approaches to each piece of the algorithm are documented. The paper also documented which of the approaches was most popular for each step of the algorithm.

From this paper, we were able to find concrete examples of state-of-the-art diarization systems and get an idea of what approaches are most popular. For instance, [3] is an example of a more typical diarization system: it uses bottom-up clustering, breaking an audio file into uniform segments and clustering these segments using GMMs, similar to our approach.

By contrast, [7] takes a top-down approach, creating an evolutive Hidden Markov Model that represents how the speakers in a particular conversation transition from one speaker to the next.

We also discovered papers on real-time diarization systems. Although we did not end up creating a real-time diarization system, we feel that a real-time system is a good next step to take once we have a robust system that can work on entire files.

Friedland and Vinyals [6], for instance, take a supervised approach to live speaker identification, where each speaker provides about 30 seconds of uninterrupted speech before proceeding with the conversation. They achieved around 85% accuracy with 1.5 seconds of latency. While their system is fairly accurate, however, the practical benefits of a real-time system are counteracted by the fact that there is a training requirement.

On the other hand, Kinnuen et. al. [5] achieve an unsupervised system with roughly 80% accuracy that can identify a speaker in less than a second. It uses vector quantization rather than the popular GMM-based approach, as well as some smart computational speed-up methods such as confidence-based speaker pruning. It's an example of what kind of results can be achieve when great amounts of work are put into choosing the best algorithms for the job. For our purposes, however, it is outside of what we could achieve in the scope of time given, and so is more of a source of inspiration for future work.

Figure 2 provides a general framework for a bottom-up diarization algorithm. Variations, of course, can exist, and these steps also don't take the specific implementation algorithms into account. As previously mentioned, the different approaches can vary widely.

3. IMPLEMENTATION

Our implementation will be a bottom-up approach that uses some classification technique to determine which speech segment belongs to which speaker. We will examine the use of both supervised and unsupervised approaches.

3.1 Datasets

Our dataset of choice is the International Computer Science Institute (ICSI) Meeting Speech data corpus (https://catalog.ldc.upenn.edu/LDC2004S02), a collection of recordings of 75 meetings. These meetings totaled roughly 72 hours and had a sample rate of 16000 Hz. The meeting audio is recorded in a special .sph format, but was converted to a .wav file before being used in our system. Each meeting also provides a hand-annotated transcript with timestamps specifying when each speaker is speaking. From this transcript, we can identify non-speech, speech with specific speakers, and overlapping speech. We focused mainly on the parts with speech and no overlapping speakers.

3.2 Algorithm

3.2.1 Data Preprocessing

We divided our non-overlapping speech data into intervals, with our interval size being 1024 samples, or roughly 0.06 seconds. After this preprocessing, each audio file had roughly between 12000 and 55000 intervals of speech and non-overlapping speakers data (since the amount of data in each audio file being composed of overlapping speakers varied widely, from 77% to 4%).

3.2.2 Features

The following main features were tested individually and combined: loudness, energy, MFCC, and spectral flatness. To get these features, we used a feature extraction tool called Yet Another Audio Feature Extractor (Yaafe). [2]

Loudness

Loudness is defined in our models as the energy in each Bark Scale Critical Band, normalized by the overall sum. Bark Scale Critical Bands are 24 frequency bands, ranging from 20 Hz to 15500 Hz. Loudness is dependent on both energy and frequency. Loudness was selected because some people talk with an energy at certain frequencies that are different from others. Of course, a person's loudness can change and could depend on emotion (i.e. when a person is excited, his/her loudness may change).

Energy, Mel Frequency Cepstral Coefficients (MFCC), Spectral Flatness

Energy is defined in our models as the root mean square of the samples in the audio frame. Mel Frequency Cepstral Coefficients (MFCC) are coefficients that are built from Mel-scale filters being applied to frequencies. The Mel-scale determines from the actual frequency the frequency that humans perceive (also known as pitch). In our feature extraction, 13 MFCC coefficients are used. Spectral flatness is calculated in our models as the ratio between the geometric and arithmetic mean of the power spectrum. Power spectrum is defined as the square of the Fourier transform of the signal. Energy, MFCC, and spectral flatness were selected because they have been shown to perform well relative to other features in past papers [1].

Combined

We also looked at the feature where we stacked loudness, energy, MFCC, and spectral flatness features in an array to experiment with combining features.

3.2.3 Number of Speakers Algorithm

Number of Speakers

Determining the number of speakers was attempted using Schwarz's Bayesian Information Criterion (BIC). BIC is used for model selection and has a penalty term to lower the possibility of the model overfitting. BIC is calculated as

$$BIC = -2 * loglikelihood + d * log(N)$$

where *loglikelihood* is the maximum log likelihood of the model, d is the number of parameters, N is the number of data points, and $d * log(N)$ is the penalty term. The smaller the BIC, the better the model [9].

Unfortunately, model selection with BIC still tended to overfit, predicting over 50 clusters consistently for each of the audio files, which only had at most 10 speakers.

3.2.4 Unsupervised Learning on Speaker Tagging Non-Overlap Audio

For unsupervised learning, both KMeans and Gaussian Mixture Models (GMM) were used. Note that for unsupervised learning, labels given by our models did not always necessarily match up with labels given in the data set, so frequency mapping was used. Basically, the cluster with the most data points was matched with the data that had the label that appeared most frequently, and so on.

KMeans Clustering

KMeans clustering is a simple unsupervised learning algorithm that clusters data into K number of groups. Qualitatively, in this algorithm, we first randomly choose K points to be our initial centroids. Then, we group each data point in the data set with the closest centroid. The positions of the each group's centroids are recalculated, and the process repeats until the positions of the centroids converge. KMeans was chosen because it has been shown that this simple method has led to very similar performance as GMMs [8]. We used Python's scikit-learn's KMeans module.

Gaussian Mixture Models (GMMs) Gaussian Mixture Model is another unsupervised learning clustering algorithm. GMMs assume that the data points are formed from a mixture of Gaussian distributions with unknown parameters. It utilizes the Expectation-Maximization (EM) algorithm and groups data points by maximizing the posterior probability that the data point belongs in a cluster. The EM algorithm is used to estimate parameters when some of the random variables are missing. Qualitatively, the EM algorithm is an iterative process over two steps: estimating the missing data given current estimates of the parameters and observed data via conditional expectation(E-step), and maximizing the likelihood function with the new estimated data to get new estimates of the parameters (M-step). We used Python's scikit-learn's GMM module.

3.2.5 Supervised Learning on Speaker Tagging Non-Overlap Audio

For supervised learning, both Support Vector Machines (SVM) and neural networks were used. We chose these two algorithms because they have been used in past papers about speaker diarization with some success [10] [11].

Supervised learning with SVM SVM was used to learn speaker models given a subset of annotated data as a training set. By default, to find the speakers in a file, we fed in 10 minutes of non-silent, non-overlapping speech from the file to train a multiclass speaker model.

Under the hood, we are simply using SVMs with linear kernels, i.e. we are finding

$$arg\min_{w,b} \frac{1}{2}\|w\|^2$$

subject to, for any $i = 1, \ldots, n$:

$$y^{(i)}(w^\top x^{(i)} - b) \geq 1$$

To extend this to a multiclass classification system, we use the one-versus-rest approach, where for a set of possible class labels $\{y_1, \cdots, y_k\}$, we train k classifiers $f_1(x), \cdots, f_k(x)$, where $f_j(x)$ distinguishes whether a data point is in class j or is in any one of the other classes. We again used Python's scikit-learn's SVM module.

Supervised learning with a neural network Another approach we experimented is to learn the speaker label via neural network. The neural network we implemented is a two-layer fully connected neural network. On the hidden layer, the sigmoid function was used to determine if a neuron would fire. We use softmax for the scorer on the output layer.

This supervised learning model takes the first third of the audio file as training data and the second part as testing data. The network can be represented as:

$$\begin{aligned} h &= W_1 x + b_1 \\ z &= sigmoid(h) \\ \theta &= W_2 z + b_2 \\ p_j &= \frac{e^{\theta_j}}{\sum_j e^{\theta_j}} \end{aligned}$$

which p is a vector, representing the probability of a data point x of having each of these labels. p_j is the j-th element in p. The model will pick the highest probability and its associated label will be the predicted. We used our own implementation for neural networks.

4. RESULTS AND DISCUSSION

The accuracy of our speaker tagging is shown in Table 2 for our features and algorithms. This accuracy is calculated by the following formula:

$$\frac{number\ of\ intervals\ that\ are\ correctly\ classified\ to\ a\ speaker}{total\ number\ of\ intervals}$$

We can see that KMeans seems to work best on energy, while GMM works best on spectral flatness. Taking into account their standard deviations, KMeans and GMM's performances overall are quite similar, as was expected.

In the SVM linear classification model, we analyzed the bias and variance for how well the data is fit into the model. In general, a high variance value may fit the training data set very well, but possibly overfit the training set and result in some noisy patterns. High bias value indicates low risk of overfitting, but might also represent underfitting the training data.

Table 1: Bias/Variance analysis of SVM Linear Classification

C parameter	0.2	0.5	1.0	1.5	2.0	2.5
$Bias^2$	1.17	1.112	1.127	1.141	1.163	1.111
Variance	0.149	0.137	0.143	0.139	0.144	0.139

Table 2: Results of the Speaker Hit Rate of the Models with ICSI Meetings

Algorithm	Features	Max	Min	Median	Mean	Std. Dev.
KMeans	Loudness	0.846	0.156	0.380	0.408	0.126
	MFCC	0.815	0.184	0.391	0.403	0.120
	Energy	0.917	0.0978	0.423	0.455	0.153
	Spectral Flatness	0.867	0.178	0.365	0.384	0.125
	Combined	0.809	0.201	0.385	0.384	0.125
GMM	Loudness	0.875	0.202	0.349	0.374	0.120
	MFCC	0.864	0.164	0.358	0.395	0.125
	Energy	0.929	0.0867	0.434	0.458	0.151
	Spectral Flatness	0.918	0.177	0.445	0.467	0.143
	Combined	0.830	0.160	0.393	0.403	0.122
Neural Net	Loudness	0.812	0.391	0.588	0.576	0.133
	MFCC	0.812	0.374	0.513	0.566	0.136
	Energy	0.812	0.381	0.570	0.536	0.134
	Spectral Flatness	0.814	0.351	0.536	0.566	0.134
	Combined	0.813	0.385	0.557	0.578	0.135
SVM	Loudness	0.986	0.196	0.507	0.542	0.164
	MFCC	0.986	0.204	0.505	0.535	0.162
	Energy	0.986	0.199	0.520	0.555	0.162
	Spectral Flatness	0.986	0.223	0.505	0.547	0.163
	Combined	0.986	0.189	0.502	0.532	0.162

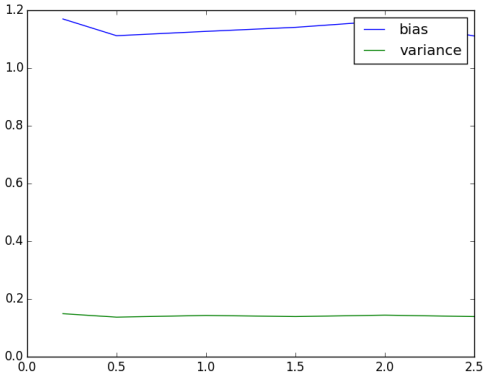

Figure 2: The plot of squared bias and variance against C parameter

From Figure 4 and Table 2 we can see that the bias reaches a peak when C is around 2.0 and the negative peak when C is around 0.5, while variance remains about the same value as C changes. Recalling that mean square error is $Bias^2 + Variance + Irreducible\ error$, we choose the point where this sum is minimal. In this case, we can choose 0.5. The reason behind the near constant value of these two values may be due to a large, fairly homogeneous data set.

For the neural network, we can observe whether the training is over-fitted or under-fitted by comparing the training accuracy with the testing. Generally, when the training accuracy is too high and much higher than the testing accuracy, it is a sign of over-fitting.

Table 3: Mean training/testing accuracy of Neural Network model

–	Loudness	MFCC	Energy	Spec Mag	Combined
train	0.568	0.556	0.566	0.564	0.549
test	0.576	0.566	0.575	0.566	0.578

In Table 3, we can see that the training accuracy is slightly lower, and yet still in the ballpark of the test accuracy.

The model for each feature is tuned separately. Since the size of each data point is different, (loudness has 24 elements, MFCC has 13, and spectral flatness and energy have only one) the depth and size of the model is also different. The size of the hidden layer is 10 for spectral magnitude and loudness and yet 20 for MFCC. The learning rate is 0.001 and the regularization is 0.0000001 for all the models. All models are trained on the training set for 100 iterations.

5. CONCLUSION, OUTLOOK, AND FUTURE WORK

Speaker tracking is a rather broad topic with many interesting and open-ended problems, ranging from just determining the number of speakers to solving the problem of speaker overlap, which is still an open problem in the field. We tried to at least touch on some of these problems, with a main focus on tagging the speakers in non-overlapping conversations. For unsupervised learning, GMMs tend to generally slightly outperform KMeans. GMM on spectral flatness seems to perform the best for our models of unsupervised learning. Supervised learning on average performs better than unsupervised learning, which is expected since the unsupervising models have no prior information except for the number of speakers. On average, neural nets perform better than SVM, though in certain situations SVM can perform extremely well, nearly flawlessly classifying non-overlapping speech segments.

Unfortunately, our methods are not as robust as we would have hoped. It appears that they can perform quite well on some audio files but also perform relatively poorly on other audio files that are from the same location and obtained using the same method. We would like to better understand what causes our methods to perform so differently.

Our implementation does not run in real-time and runs on recorded data. In the future, we can use vector quantization to shorten processing time. Vector quantization is a form of approximation that maps an infinite set of vectors to a finite

set of vectors, which can aid in real-time analysis. However, a major problem with our system, as previously mentioned, is when there are multiple speakers talking simultaneously. This was significant in our data set, as overlapping speakers composed of around 21% on average of each audio file. There are at least three major parts to this problem: determining when speaker overlap occurs, determining the number of speakers in the overlap, and actually extracting the data for each of the overlapping speakers. One approach would be to apply a diarization module where the wave characteristics of several speakers talking together is different than the wave characteristics of one speaker [4].

6. REFERENCES

[1] Xavier Anguera Miro. Robust, *Speaker Diarization for Meetings. Speech Processing Group. Department of Signal Theory and Communications. Universitat Polit'ecnica de Catalunya. Barcelona, October 2006*

[2] YAAFE, *an Easy to Use and Efficient Audio Feature Extraction Software, B.Mathieu, S.Essid, T.Fillon, J.Prado, G.Richard, proceedings of the 11th ISMIR conference, Utrecht, Netherlands, 2010.*

[3] M. Ben, M. Betser, F. Bimbot, and G. Gravier, *"Speaker Diarization using Bottom-up Clustering based on a Parameter-derived Distance between Adapted GMMs," in Proc. Int. Conference on Spoken Language Processing (ICSLP), Jeju Island, Korea, October 2004.*

[4] Rao, K. Sreenivasa, and Sourjya Sarkar. *"Robust Speaker Verification: A Review." SpringerBriefs in Electrical and Computer Engineering Robust Speaker Recognition in Noisy Environments (2014)*: 13-27. Web.

[5] Kinnunen, T., E. Karpov, and P. Franti. *"Real-time Speaker Identification and Verification."* IEEE Transactions on Audio, Speech and Language Processing IEEE Trans. Audio Speech Lang. Process. 14.1 (2006): 277-88. Web.

[6] Friedland, Gerald, and Oriol Vinyals. *"Live Speaker Identification in Conversations."* Proceeding of the 16th ACM International Conference on Multimedia - MM '08 (2008): Web.

[7] Vallet, Félicien, Slim Essid, and Jean Carrive. *"A Multimodal Approach to Speaker Diarization on TV Talk-Shows."* IEEE Trans. Multimedia IEEE Transactions on Multimedia 15.3 (2013): 509-20. Web.

[8] Anguera, Xavier., 2006. *Robust Speaker Diarization for Meetings*, Ph.D. thesis, Universitat Politecnica de Catalunya,.

[9] Schwarz, G. (1978). *Estimating the dimension of a model.* Annals of Statistics 6(2), pp. 461-464.

[10] A. Temko, D. Macho, and C. Nadeu, "Enhanced SVM Training for Robust Speech Activity Detection," in Proc. ICASSP, Hawai, USA, 2007.

[11] S. Jothilakshmi, V. Ramalingam, and S. Palanivel, "Speaker diarization using autoassociative neural networks," Engineering Applications of Artificial Intelligence, vol. 22(4-5), 2009.