Digital Signal Processing: Speaker Recognition
Final Report
(Complete Version)

Xinyu Zhou, Yuxin Wu, and Tiezheng Li
Tsinghua University

Contents

1 Introduction

Speaker recognition is the identification of the person who is speaking by characteristics of their voices (voice biometrics), also called voice recognition. [27]

A **Speaker Recognition** tasks can be classified with respect to different criterion: Text-dependent or Text-independent, Verification (decide whether the person is he claimed to be) or Identification (decide who the person is by its voice).[27]

Speech is a kind of complicated signal produced as a result of several transformations occurring at different levels: semantic, linguistic and acoustic. Differences in these transformations may lead to differences in the acoustic properties of the signals. The recognizability of speaker can be affected not only by the linguistic message but also the age, health, emotional state and effort level of the speaker. Background noise and performance of recording device also interfere the classification process.

Speaker recognition is an important part of Human-Computer Interaction (HCI). As the trend of employing wearable computer reveals, Voice User Interface (VUI) has been a vital part of such computer. As these devices are particularly small, they are more likely to lose and be stolen. In these scenarios, speaker recognition is not only a good HCI, but also a combination of seamless interaction with computer and security guard when the device is lost. The need of personal identity validation will become more acute in the future. Speaker verification may be essential in business telecommunications. Telephone banking and telephone reservation services will develop rapidly when secure means of authentication were available.

Also,the identity of a speaker is quite often at issue in court cases. A crime victim may have heard but not seen the perpetrator, but claim to recognize the perpetrator as someone whose voice was previously familiar; or there may be recordings of a criminal whose identity is unknown. Speaker recognition technique may bring a reliable scientific determination.

Furthermore, these techniques can be used in environment which demands high security. It can be combined with other biological metrics to form a multi-modal authentication system.

In this task, we have built a proof-of-concept text-independent speaker recognition system with GUI support. It is fast, accurate based on our tests on large corpus. And the gui program only require very short utterance to quickly respond. The whole system is fully described in this report. This project is developed at Git9[1], and is also hosted on github[2]. The repository contains the source code, all documents, experiment log, as well as a video demo. The complete pack of this project also contains all the intermediate data, models, recordings, and 3rd party libraries.

2 Algorithms

In this section we will present our aproach to tackle the speaker recognition problem.

An utterance of a user is collected during enrollment procedure. Further processing of the utterance follows following steps:

[1]Git hosting service of the department of CST, Tsinghua Univ., currently maintained by Yuxin Wu. See http://git.net9.org

[2]See https://github.com/ppwwyyxx/speaker-recognition

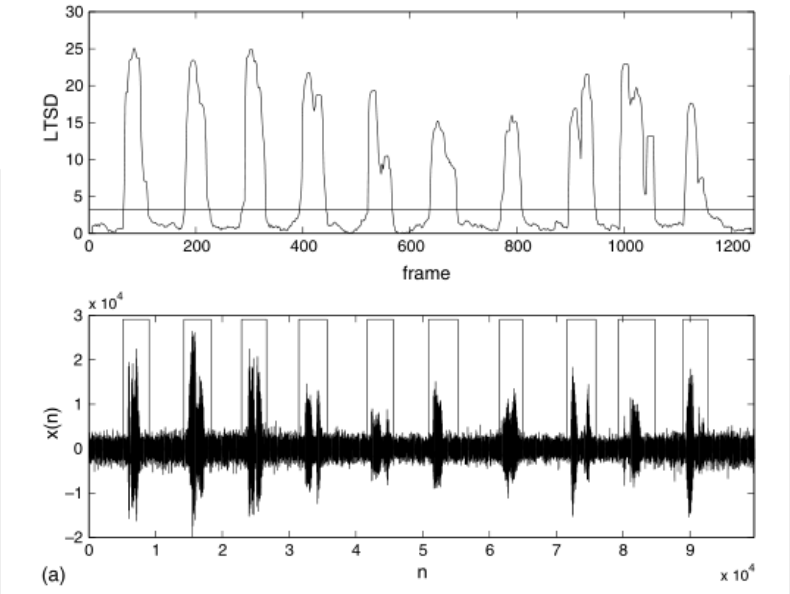
2.1 VAD

Signals must be first filtered to rule out the silence part, otherwise the training might be seriously biased. Therefore **Voice Activity Detection** must be first performed.

An observation found is that, the corpus provided is nearly noise-free. Therefore we use a simple energy-based approach to remove the silence part, by simply remove the frames that the average energy is below 0.01 times the average energy of the whole utterance.

This energy-based method is found to work well on database, but not on GUI. We use LTSD(Long-Term Spectral Divergence) [21] algorithm on GUI, as well as noise reduction technique from SOX[26] to gain better result in real-life application.

LTSD algorithm splits a utterance into overlapped frames, and give scores for each frame on the probability that there is voice activity in this frame. This probability will be accumulated to extract all the intervals with voice activity. A picture depicting the principle of LTSD is as followed:



Since this is not our primary-task, we shall not expand details here. For further information on how these works, please consult original paper.

2.2 Feature Extraction

2.2.1 MFCC

Mel-Frequency Cepstral Coefficient is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel-scale of frequency [15] . MFCC is the mostly widely used features in Automatic Speech Recognition(ASR), and it can also be applied to Speaker Recognition task.

The process to extract MFCC feature is demonstrated in Figure.1

First, the input speech should be divided into successive short-time frames of length L, neighboring frames shall have overlap R. Those frames are then windowed by Hamming Window, as shown in Figure.2.

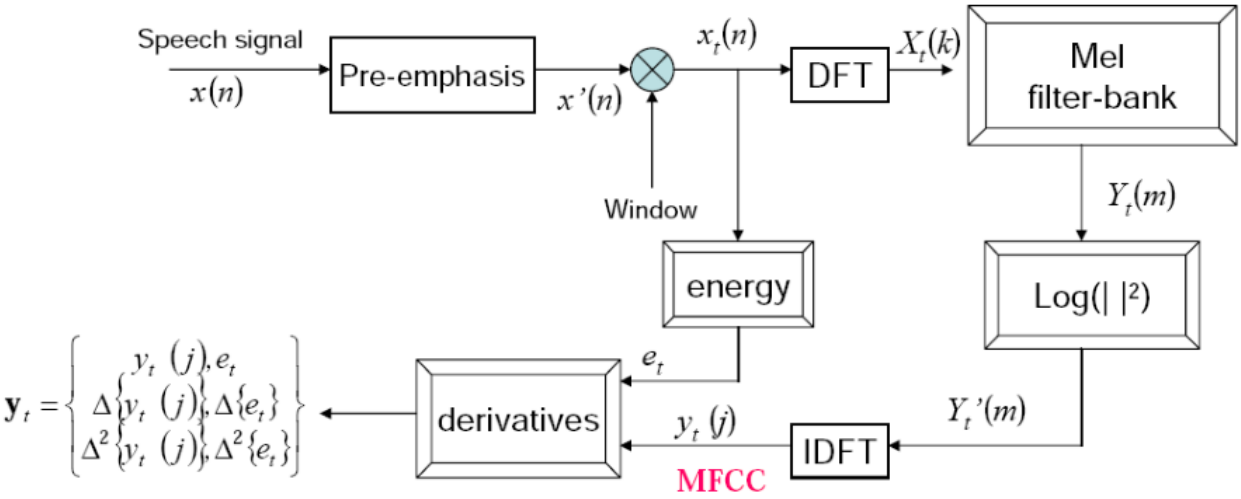


Figure 1: MFCC feature extraction process

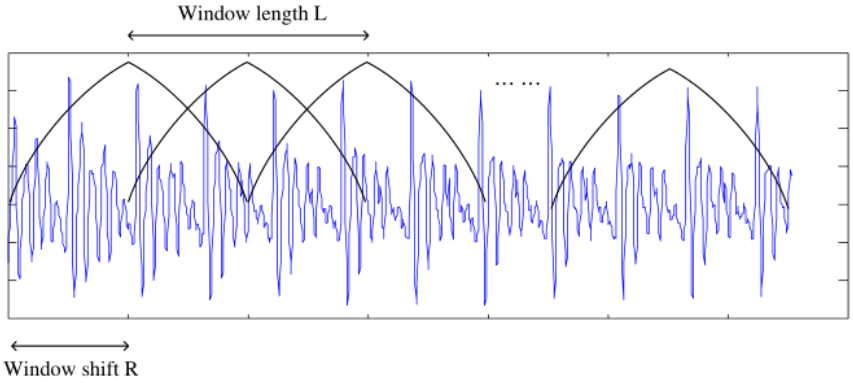


Figure 2: Framing and Windowing

Then, We perform Discrete Fourier Transform (DFT) on windowed signals to compute their spectrums. For each of N discrete frequency bands we get a complex number $X[k]$ representing magnitude and phase of that frequency component in the original signal.

Considering the fact that human hearing is not equally sensitive to all frequency bands, and especially, it has lower resolution at higher frequencies. Scaling methods like Mel-scale are aimed at scaling the frequency domain to better fit human auditory perception. They are approximately linear below 1 kHz and logarithmic above 1 kHz, as shown below in Figure.3:

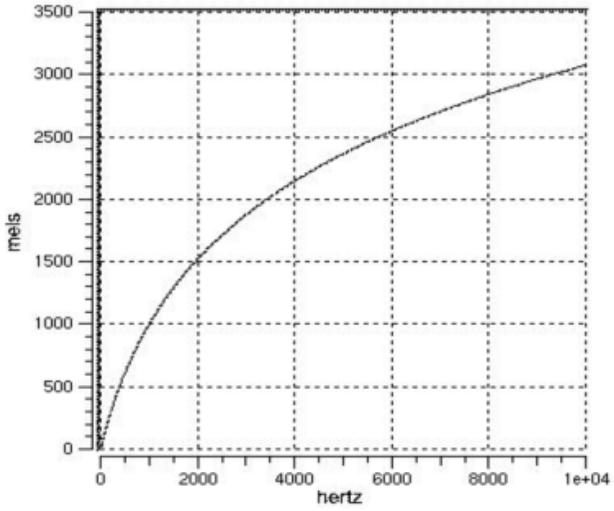


Figure 3: Mel-scale plot

In MFCC, Mel-scale is applied on the spectrums of the signals. The expression of Mel-scale warpping is as followed:

$$M(f) = 2595 \log_{10}(1 + \frac{f}{700})$$

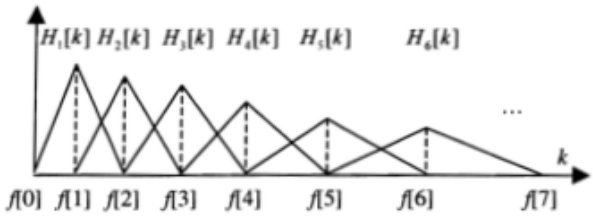


Figure 4: Filter Banks (6 filters)

Then, we appply the bank of filters according to Mel-scale on the spectrum, calculate the logarithm of energy under each bank by $E_i[m] = \log(\sum_{k=0}^{N-1} X_i[k]^2 H_m[k])$ and apply Discrete Cosine Transform (DCT) on $E_i[m](m = 1, 2, \cdots M)$ to get an array c_i:

$$c_i[n] = \sum_{m=0}^{M-1} E_i[m] \cos(\frac{\pi n}{M}(m - \frac{1}{2}))$$

Then, the first k terms in c_i can be used as features for future training. The number of k varies in different cases, we will further discuss the choice of k in Section.5.

2.2.2 LPC

Linear predictive coding is a tool used mostly in audio signal processing and speech processing for representing the spectral envelope of a digital signal of speech in compressed form, using the information of a linear predictive model.[14]

The basic assumption in LPC is that, in a short period, the nth signal is a linear combination of previous p signals: $\hat{x}(n) = \sum_{i=1}^{p} a_i x(n-i)$ Therefore, to estimate the coefficients a_i, we have to minimize the squared error $\mathrm{E}\left[\hat{x}(n) - x(n)\right]$. This optimization can be done by Levinson-Durbin algorithm.[13]

Therefore, we first split the input signal into frames, as is done in MFCC feature extraction Section.2.2.1. Then we calculate the k order LPC coefficients for the signal in this frame. Since the coefficients is a compressed description for the original audio signal, the coefficients is also a good feature for speech/speaker recognition. The choice of k will also be further discussed in Section.5.

2.3 GMM

Gaussian Mixture Model is commonly used in acoustic learning task such as speech/speaker recognition, since it describes the varied distribution of all the feature vector.[24] GMM assumes that the probability of a feature vector x belonging to the model is the following:

$$p(x|w_i, \mu_i, \Sigma_i) = \sum_{i=1}^{K} w_i \mathcal{N}(x|\mu_i, \Sigma_i) \tag{1}$$

where

$$\mathcal{N}(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{|\Sigma_i|}} \exp\left(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right)$$

subject to

$$\sum_{i=1}^{K} w_i = 1$$

Therefore, GMM is merely a weighted combination of multivariate Gaussian distribution which assumes feature vectors are independent. (Actually we use diagonal covariances since the dimensions of the feature vector is independent to each other). GMM can describe the distribution of feature vector with several clusters, as shown in Figure.5

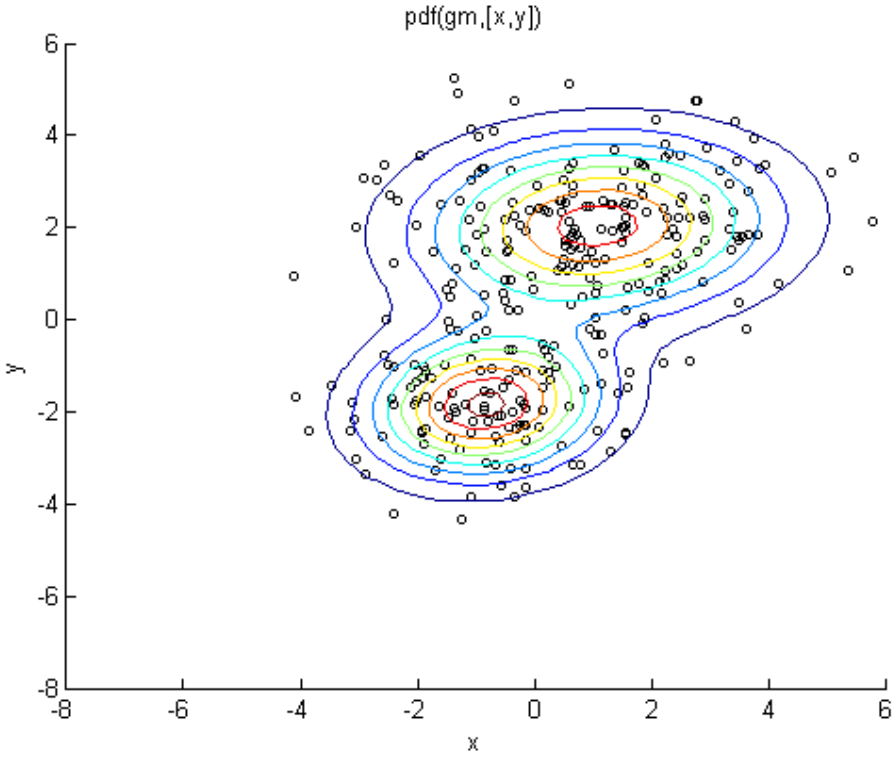


Figure 5: A Two-Dimensional GMM with Two Components

The training of GMM is the process to find the best parameters for μ_i, Σ_i, w_i, so that the model fits all the training data with maximized likelihood. More specifically, Expectation-Maximization(EM) Algorithm[4] is used to maximize the likelihood. The two steps of one iteration of the algorithm in GMM training case here are

- **E-Step** For each data point(feature vector), estimate the probatility that each Gaussian generated it[3]. This is done by direct computation using Equation.1.
- **M-Step** Modify the parameters of GMM such that maximize the likelihood of data. Here, hidden variable z_{ij} is introduced to indicate where i-th data point is generate by Gaussian j. It can be shown that, instead of maximizing the likelihood of data, we can maximize the expectation of log likehood of data with respect to Z.

  let $\theta = \{w, \theta, \Sigma\}$, the log likehood function is

  $$Q(\theta', \theta) = \mathbb{E}_Z[\log p(X, Z)|\theta]$$

  where θ is current parameters, and θ' is the parameters we are to estimate. Incorporating the constraint $\sum_{i=1}^{K} w_i = 1$ using Lagrange multiplier gives

  $$J(\theta', \theta) = Q(\theta', \theta) - \lambda \left(\sum_{i=1}^{K} w_i - 1 \right)$$

[3]Actually, for an arbitrary point in space, its measure is zero, therefore its "probability" is actually zero. Therefore, here by "probability of x" we mean "the value of probability distribution function at x"

Set derivatives to zero, we can get the update equation

$$Pr(i|x_j) = \frac{w_i \mathcal{N}(x_j|\mu_i', \Sigma_i')}{\sum_{k=1}^{K} w_k \mathcal{N}(x_j|\mu_k' \Sigma_k')}$$

$$n_i = \sum_{j=1}^{N} Pr(i|x_j)$$

$$\mu_i = \frac{1}{n_i} \sum_{t=1}^{T} Pr(i|x_j) x_j$$

$$\Sigma_i = \left(\frac{1}{n_i} \sum_{t=1}^{T} Pr(i|x_j) diag(x_j x_j^T) \right) - diag(\mu_i' \mu_i'^T)$$

$$w_i = \frac{n_i}{N}$$

After training, the model can give the score of fitness for every input feature vector, measuring the probability that the vector belongs to this model.

Therefore, in the task of speaker recognition, we can train a GMM for every speaker. Then for a input signal, we extract lists of feature vectors for it, and calculate the overall likelihood that the vectors belong to each model. The speaker whose model fits the input best will be choosen as the answer.

Moreover, an enhancement have been done to the original GMM method. The training of GMM first requires a random initialization of the means of all the components. However, we can first use K-Means algorithm[12] to perform a clustering to all the vectors, then use the clustered centers to initialize the training of GMM. This enhancement can speed up the training, also gives a better training result.

On the calculation of K-Means, an algorithm call K-MeansII[3], which is an improved version of K-Means++ [2] can be used for better accuracy.

2.4 UBM

Universal Background Model is a GMM trained on giant number of speakers. It therefore describes common acoustic features of human voices.[30]

As we are providing continuous speech close-set diarization function in GUI, we adopt **Universal Background Model** as imposter model using equation given in [23] and use likelihood ratio test to make reject decisions as proposed in[23].

Further more, by hints mentioned in paper, we only update mean vectors.

When using conversation mode in GUI (will be present later), GMM model of each user is adapted from a pre-trained UBM using method described in [23].

2.5 CRBM

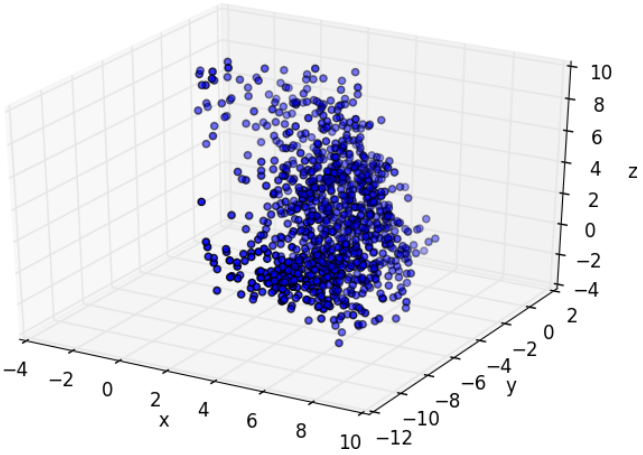
Restricted Boltzmann Machine is generative stochastic two-layer neural network that can learn a probability distribution over its set of binary inputs[22]. **Continuous restricted Boltzmann Machine(CRBM)**[5] extends its ability to real-valued inputs. RBM has a ability to, given an input(visible layer), reconstruct a hidden

layer that is similar to the input. The neurons in hidden layer controls the model complexity and the performance of the network. The Gibbs sampling of hidden layer can be seen as a representation of the original data. Therefore RBMs can be used as an auto feature-extractor. Figure.6 illustrate original MFCC data and the sampled output of reconstructed data from CRBM.

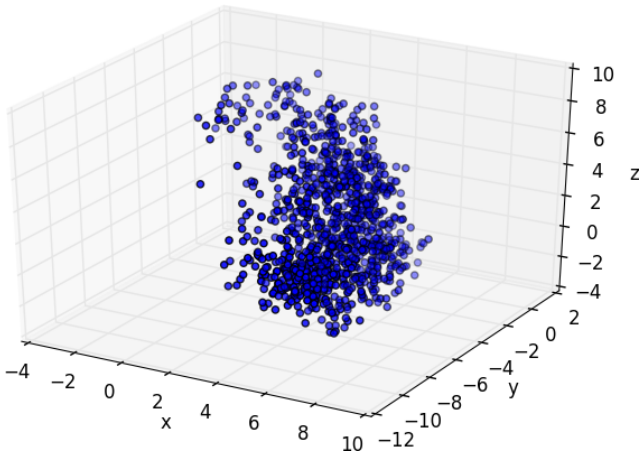
Both RBM and CRBM can be trained using Contrastive Divergence learning, with subtle difference in update equation.

As details about CRBM are too verbose to be covered here, for interested, we recommend reading original papers.

Previous works using neural network largely focused on speech recognition, such as [6],[16].



The first three dimension of a woman's MFCC feature



The first three dimension of the same woman's MFCC feature recontructed by a CRBM with 50-neuron hidden layer. We can see that, the density of these two distributions are alike

Figure 6

To use CRBM as a substitution of GMM, rather than an feature extractor, we train a CRBM per speaker, and estimate reconstruction error without sampling (which is stable). The person whose corresponding CRBM has lowest reconstruction error is chosen as

recognition result.

2.6 JFA

Factor Analysis is a typical method which behave very well in classification problems, due to its ability to account for different types of variability in training data. Within all the factor analysis methods, Joint Factor Analysis (JFA)[11, 9] was proved to outperform other method in the task of Speaker Recognition.

JFA models the user by "supervector" , i.e. a $C \times F$ dimension vector, where C is the number of components in the Universal Background Model, trained by GMM on all the training data, and F is the dimension of the acoustic feature vector. The supervector of an utterance is obtained by concatenate all the C means vectors in the trained GMM model. The basic assumption of JFA on describing a supervector is:

$$\vec{M} = \vec{m} + vy + dz + ux,$$

where m is a supervector usually selected to be the one trained from UBM, v is a $CF \times R_s$ dimension matrix, u is a $CF \times R_c$ dimension matrix, and d is a diagonal matrix. This four variables are considered independent of all kinds of variabilities and remain constant after training, and x, y, z are matrixes computed for each utterance sample. In this formulation, $m + vy + dz$ is commonly believed to account for the "Inter-Speaker Variability", and ux accounts for the "Inter-Channel Variability". The parameter R_s and R_c, also referred to as "Speaker Rank" and "Channel Rank", are two emprical constant selected as first. The training of JFA is to calculate the best u, v, d to fit all the training data.

3 Implementation

The whole system is written mainly in python, together with some code in C++ and matlab. The system strongly relies on the support of the numpy[17] and scipy[25] library.

1. VAD

   Three types of VAD filters are located in `src/filters/`.

   `silence.py` implements an energy-based VAD algorithm. `ltsd.py` is a wrapper for LTSD algorithm, relying on pyssp[20]. `noisered.py` is a wrapper for SOX noise reduction tools, relying on SOX [26] being installed in the system.

2. Feature

   Implementations for feature extraction are locaed in `src/feature/`.

   `MFCC.py` is a self-implemented MFCC feature extractor. `BOB.py` is a wrapper for the MFCC feature extraction in the bob [1] library. `LPC.py` is a LPC feature extractor, relying on `scikits.talkbox` [28]. All the three extractor have the same interface, with configurable parameters.

   In the implemention, we have tried different parameters of these features. The test script can be found as `src/test/test-feature.py` According to our experiments, we have found that the following parameters are optimal:

   - Common parameters:
     - Frame size: 32ms
     - Frame shift: 16ms
     - Preemphasis coefficient: 0.95
   - MFCC parameters:
     - number of cepstral coefficient: 15
     - number of filter banks: 55
     - maximal frequency of the filter bank: 6000
   - LPC Parameters:

- number of coefficient: 23

3. GMM

   We have tried GMM from scikit-learn [18] as well as pypr [31], but they suffer a common problem of inefficency. For the consideration of speed, a C++ version of GMM with K-MeansII initialization and concurrency support was implemented and located in `src/gmm/`. It requires `g++>=4.7` to compile. This implementation of GMM also provides a python binding which have similar interface to the GMM in scikit-learn.

   The new version of GMM, has enhancement in both speed and accuracy. A more detailed discussion will be in Section.5.

   At last, we used GMM with 32 components, which is found to be optimal according to our experiment. The covariance matrix of every Gaussian component is assumed to be diagonal, since each dimension of the feature vector are independent.

4. CRBM

   CRBM is implemented in C++, located in `src/nn`. It also has concurrency support.

5. JFA

   From our investigation, we found that the original algorithm [9] for training JFA model is of too much complication and hard to implement. Therefore, we use the simpler algorithm presented in [10] to train the JFA model. This JFA implementation is based on JFA cookbook[8]. To generate feature files for JFA, `test/gen-features-file.py` shall be used. After `train.lst, test.lst, enroll.lst` are properly located in `jfa/feature-data`, the script `run_all.m` will do the training and testing, and `exp/gen_result.py` will calculate the accuracy.

   However, from the result, JFA does not seem to outperform our enhanced MFCC and GMM algorithms (but do outperform our old algorithms). It is suspected that the training of a JFA model needs more data than we have provided, since JFA needs data from various source to account for different types of variabilities. Therefore, we might need to add extra data on the training of JFA, but keep the same data scale in the stage of enrollment, to get a better result.

   It is also worth mentioning that the training of JFA will take much longer time than our old method, since the estimation process of u, v, d does not converge quickly. As a result, it might not be practical to add JFA approach to our GUI system. But we will still test further on the performance of it, compared to other methods.

6. GUI

   GUI is implemented based on PyQt[29] and PyAudio[19]. `gui.py` is the entrance point. The usage of GUI will be introduced in Section.6.

4 Dataset

In the filed of speech/speaker recognition, there are some research oriented corpus, but most of them are expensive. [7] gives a detailed list on the popular speech corpus for speech/speaker recognition. In this system, we mainly use the speech corpus provided by our teacher Xu.

The dataset provided comprised of 102 speaker, in which 60 are females and the rest are males. The dataset contains three different speaking style: Spontaneous, Reading and Whisper. Some simple statistics are as follows:

	Spontaneous	Reading	Whisper
Average Duration	202s	205s	221s
Female Average Duration	205s	202s	217s
Male Average Duration	200s	203s	223s

5 Performance

We have tested our approaches under various parameters, based on a corpus described in Section.4.

All the tests in this section have been conducted serval times (depending on computation cost, vary from 10 to 30) with random selected training and testing speakers. The average over these tests are considered as confidential result.

5.1 Efficiency Test of our GMM

We have extensively examined the efficiency of our implementation of GMM compared to scikit-learn version. Test is conducted using real MFCC data with 13 dimensions, 20ms frame length. We consider the scenario when training a UBM with 256 mixtures. We examine the time used for ten iteration. For comparable results, we diabled the K-means initialization process of both scikit-learn GMM implementation and ours. Time used for ten iterations under different data size and concurrency is recorded.

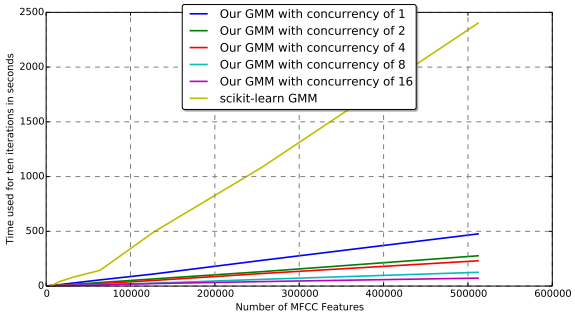


Figure 7: Comparison on efficiency

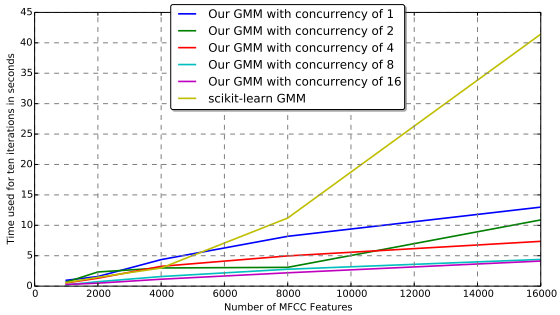


Figure 8: Comparison on efficiency when number of MFCC features is small

From Figure.7, we can immediately infer that our method is much-much more efficient than the widely used version of GMM provided by scikit-learn when the data size grows sufficiently large.

We shall analyze in two aspect:

- No concurrency

  When the number of MFCC features grows sufficiently large, our method shows great improvement. When training 512,000 features, our method is 5 times faster than comparing method.

- With concurrency

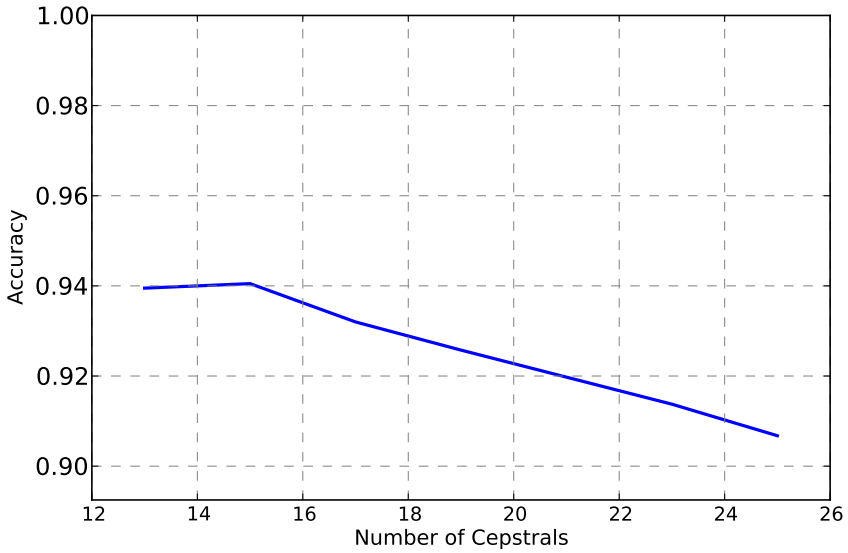
  Our method shows considerable concurrency scalability that the running time is approximately lineary to the number of cores using.

  When using 8-cores, our method is 19 **times** faster than comparing method.

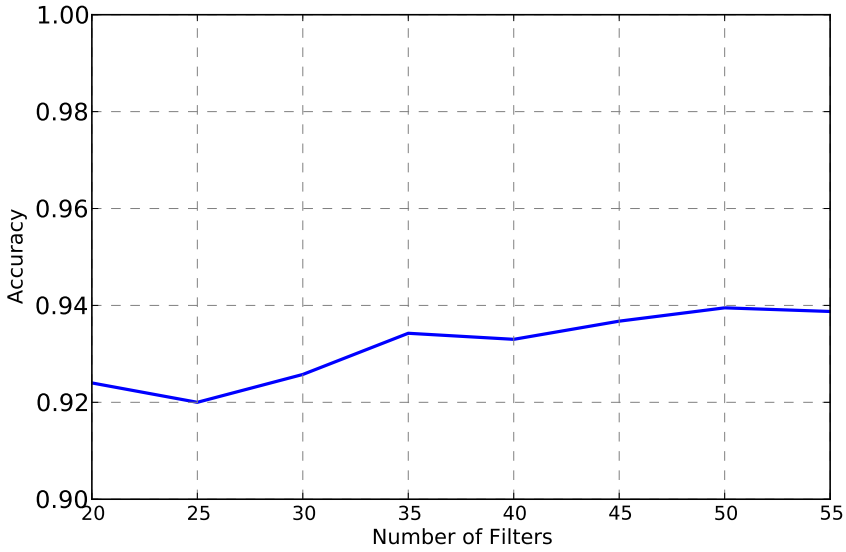
5.2 Change in MFCC Parameters

The following tests reveal the effect of MFCC parameters on the final accuracy. The tests were all performed on "Style-Reading" corpus with 40 speakers, each with 20 seconds for enrollment and 5 seconds for recognition.

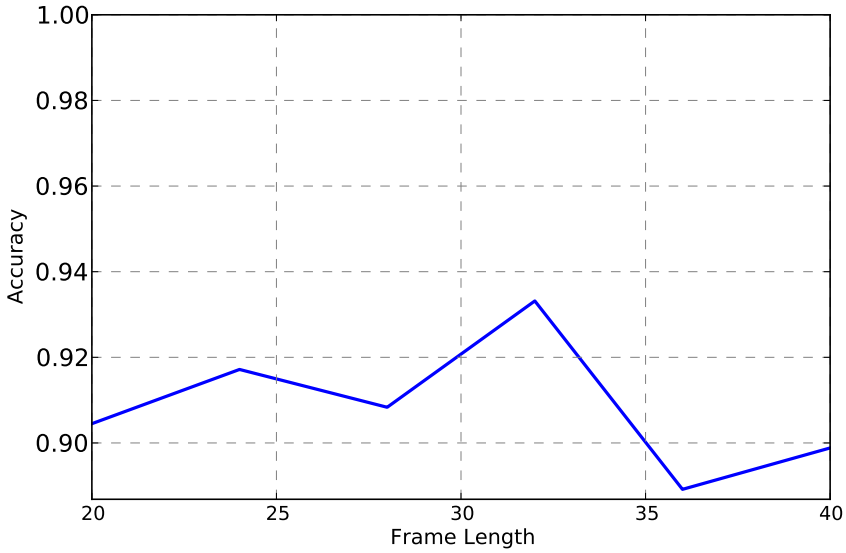
1. Different Number of Cepstrums



2. Different Number of Filterbanks



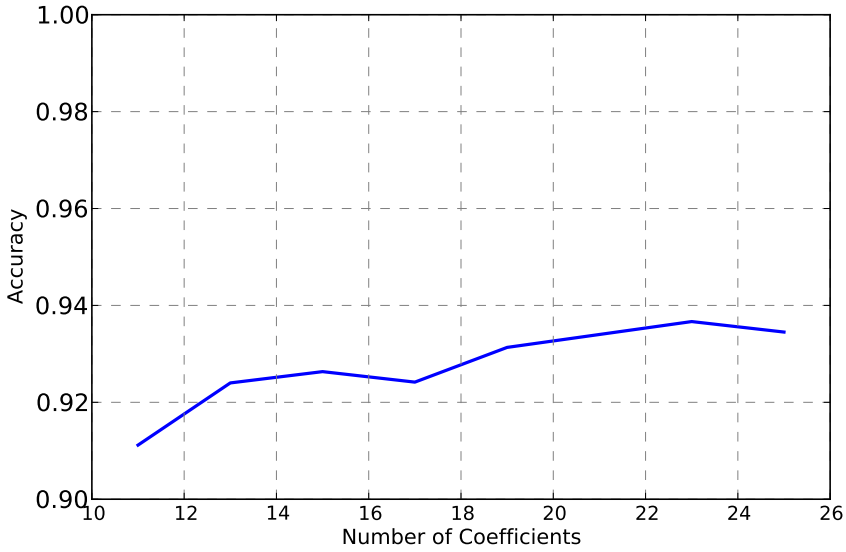
3. Different Size of Frame



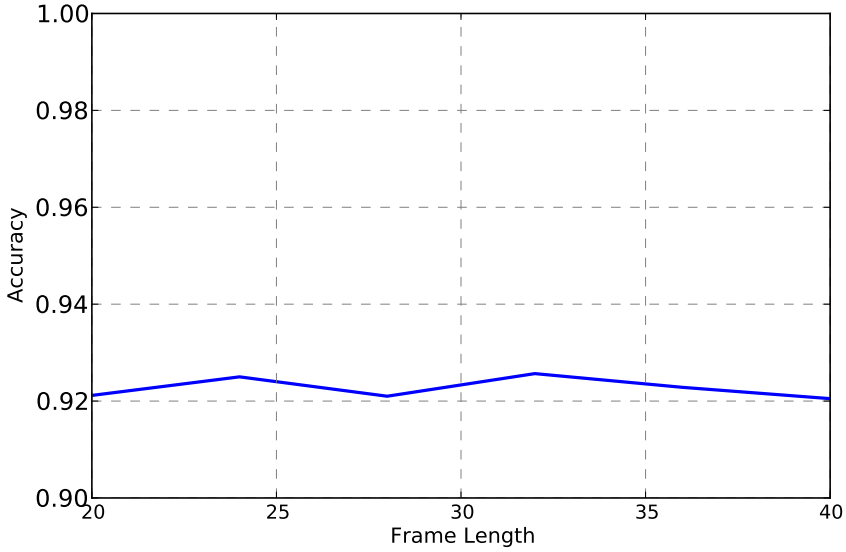
5.3 Change in LPC Parameters

The following tests display the effect of LPC parameters on the final accuracy. The tests were performed on "Style-Reading" with 40 speakers, each with 20 seconds for enrollment and 5 seconds for recognition.

1. Different Number of Coefficient

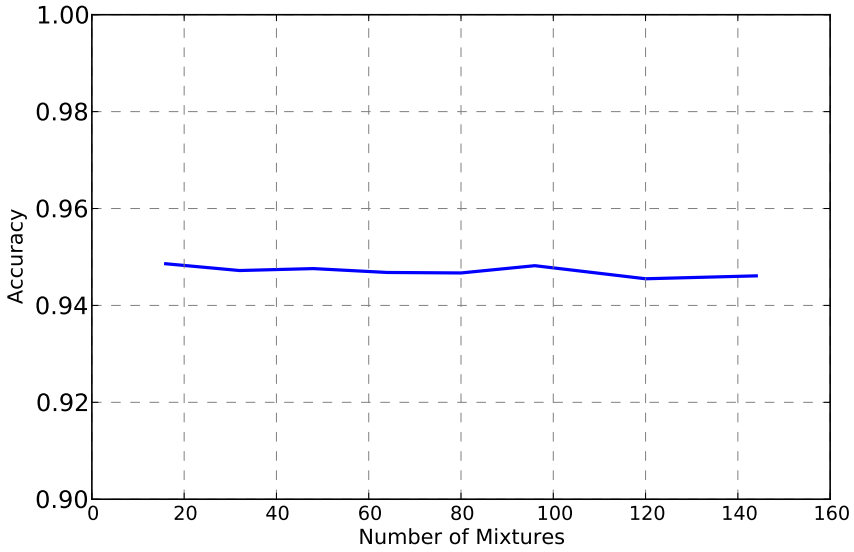


2. Different Size of Frame



5.4 Change in GMM Components

We experimented on the effect of GMM Components. We found that the number of components have slight effect on the accuracy, but a GMM with higher order might take significantly longer time to train. Therefore we still use GMM with 32 components in our system.



5.5 Different GMM Algorithms

We compare our implementation of GMM to GMM in scikits-learn.

The configurations of the test is as followed:

- Only MFCC: frame size is $20ms$, 19 cepstrums, 40 filterbanks
- Number of mixtures is set to 32, the optimal number we found previously
- GMM from scikit-learn, compared to our GMM.
- 30s training utterance and 5s test utterance
- 100 sampled test utterance for each user

From this graph we could see that, our GMM performs better than GMM from scikit-learn in general. Due to the random selection of test data, the variance of the test can be high when the number of speakers is small, as is also the case in the next experiment. But this result still shows that our optimization on GMM takes effect.

5.6 Accuracy Curve on Different Number of Speakers

An apparent trade-off in speaker recognition task is the number of speakers enrolled and the accuracy on recognization. Also, the duration of signal for enrollment and test can have significant effect on the accuracy. We've conducted test using well-tuned parameters for feature extraction as well as GMM, on dataset with various number of people and with various test duration.

The configurations of this experiment is as followed:

- Database: "Style-Reading"

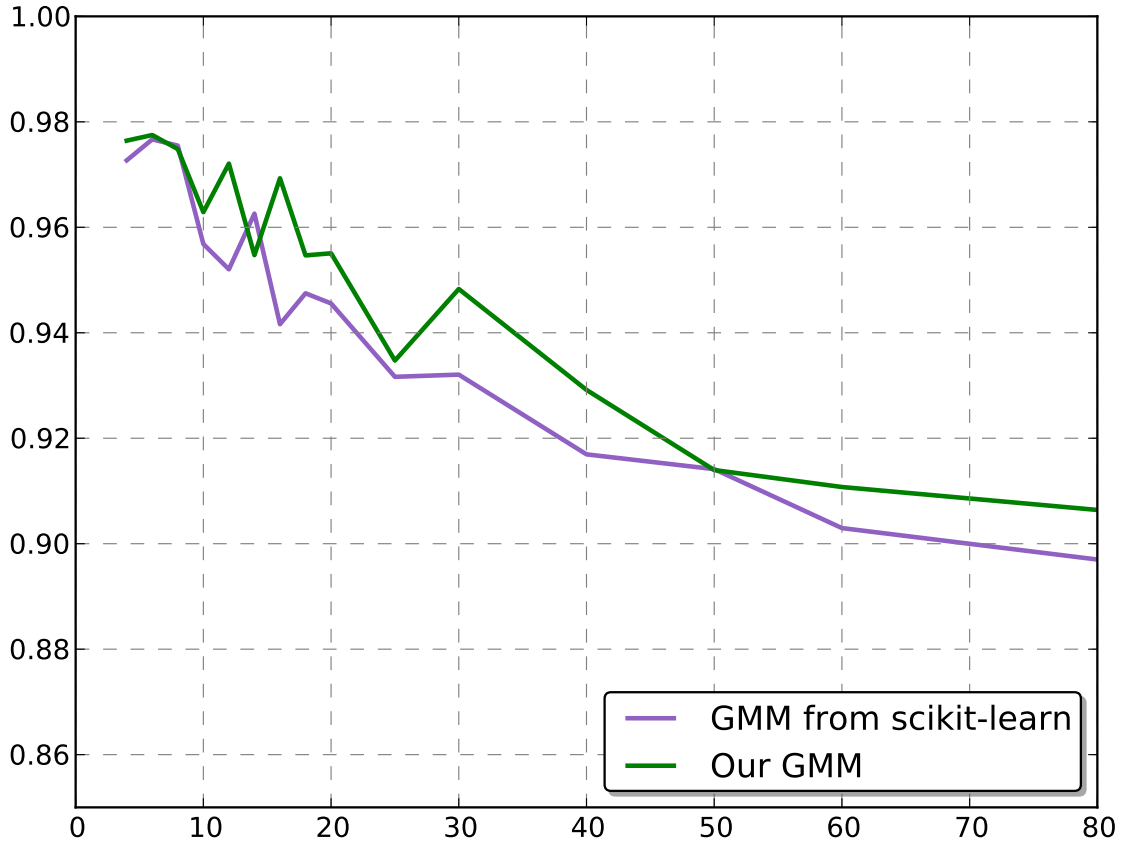
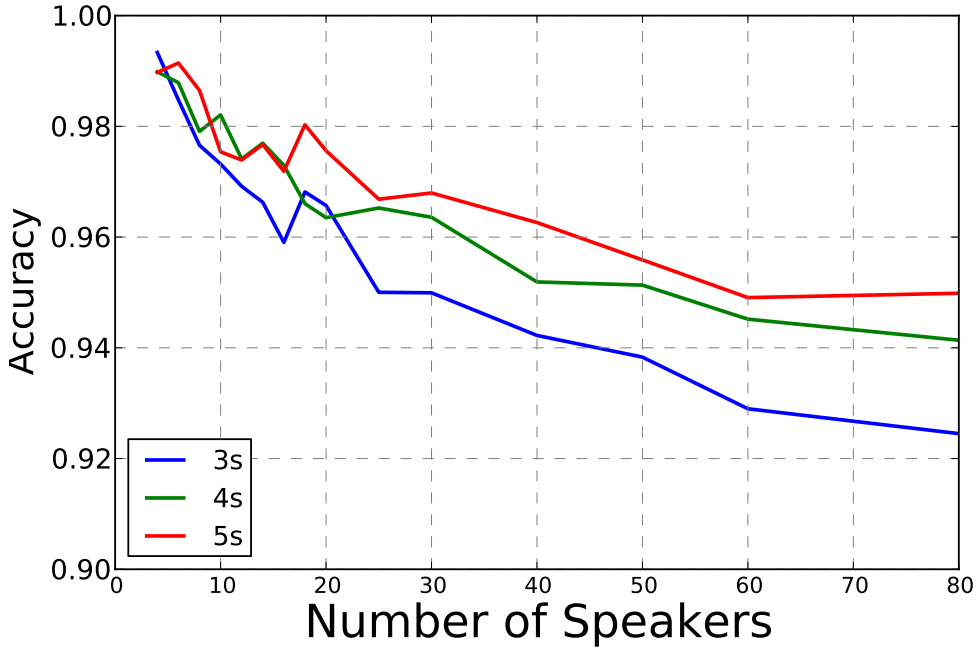


Figure 9: Accuracy curve for two GMM

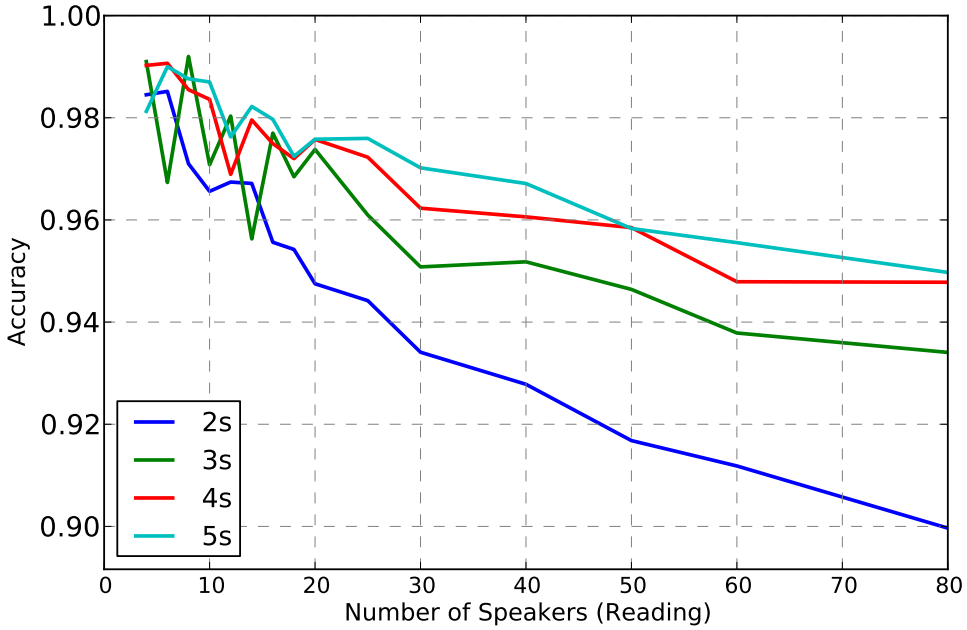
- MFCC: frame size is $32ms$, 19 cepstrums, 55 filterbanks
- LPC: frame size is $32ms$, 15 coefficients
- GMM from scikit-learn, number of mixtures is 32
- 20s utterance for enrollment
- 50 sampled test utterance for each user

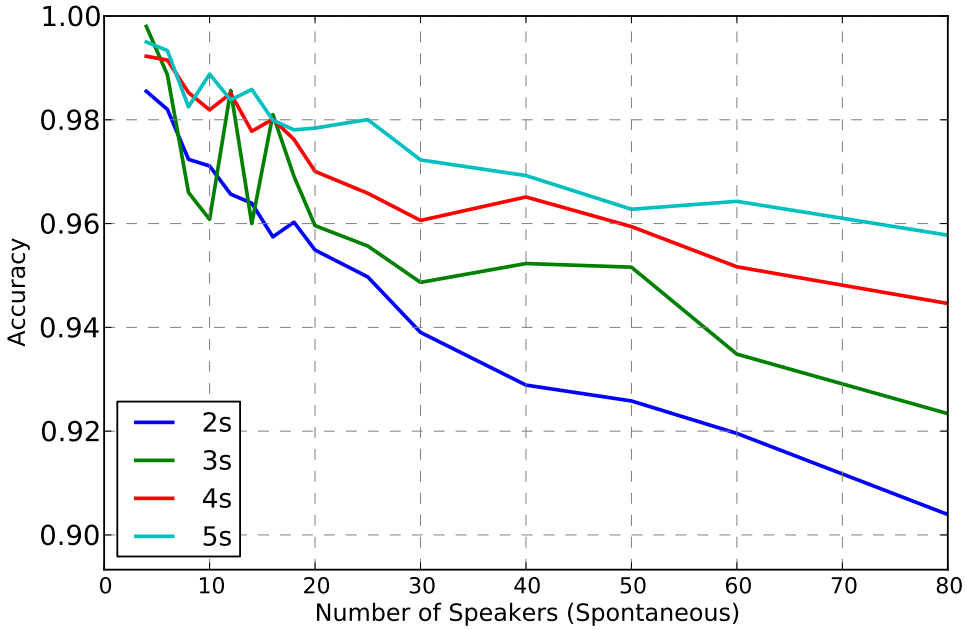


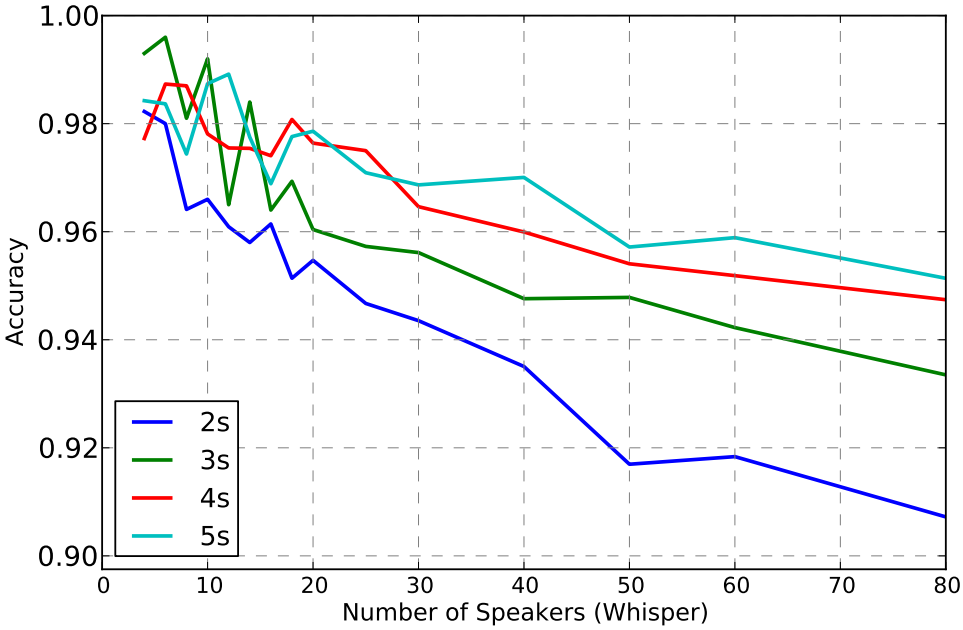
We also conducted experiments on different style of corpus. The configurations of this experiment is as followed:

- MFCC: frame size is $32ms$, 15 cepstrums, 55 filterbanks
- LPC: frame size is $32ms$, 23 coefficients
- GMM from scikit-learn, number of mixtures is 32
- 20s utterance for enrollment
- 50 sampled test utterance for each user

The result is shown below. Note that each point in the graph is an average value of 20 independent test with random sampled speakers .


1.00
0.98
0.96
0.94
0.92
0.90
Accuracy
2s
3s
4s
5s
0
10
20
30
40
50
60
70
80
Number of Speakers (Reading)


1.00
0.98
0.96
0.94
0.92
0.90
Accuracy
2s
3s
4s
5s
0
10
20
30
40
50
60
70
80
Number of Speakers (Spontaneous)



5.7 CRBM Performance Test

We also tested RBM using following configuration:

- MFCC: frame size is $32ms$, 15 cepstrums, 55 filterbanks
- LPC: frame size is $32ms$, 23 coefficients
- CRBM with 32 hidden units.
- 50 sampled test utterance for each user
- 5s test utterance

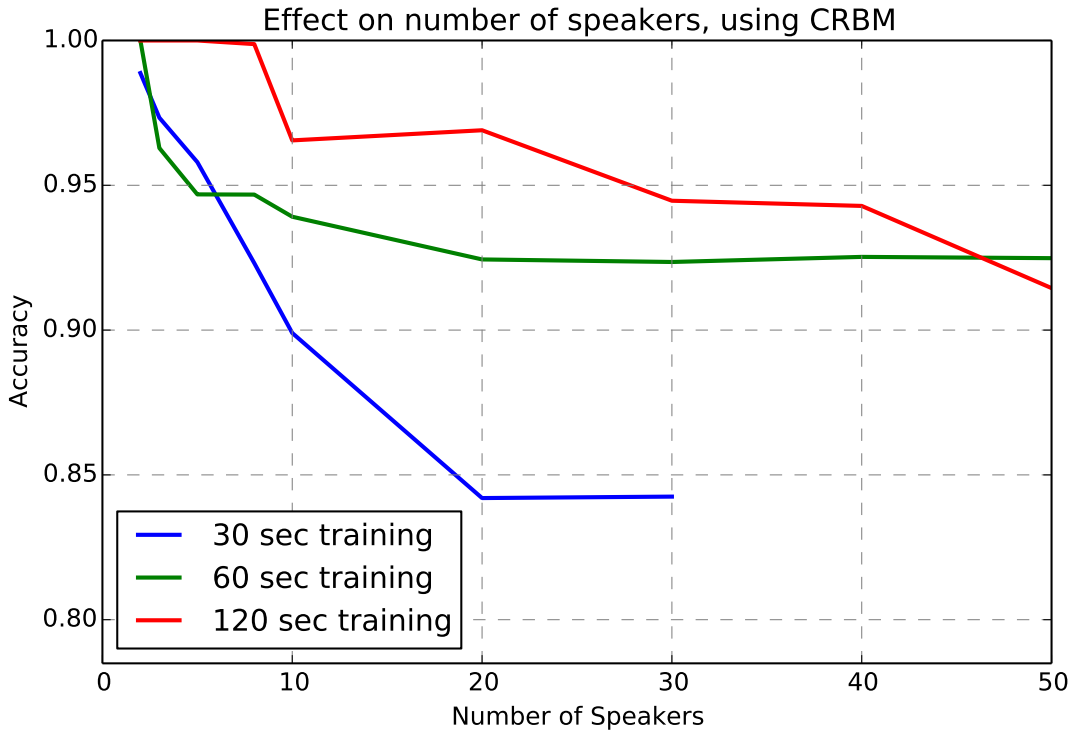


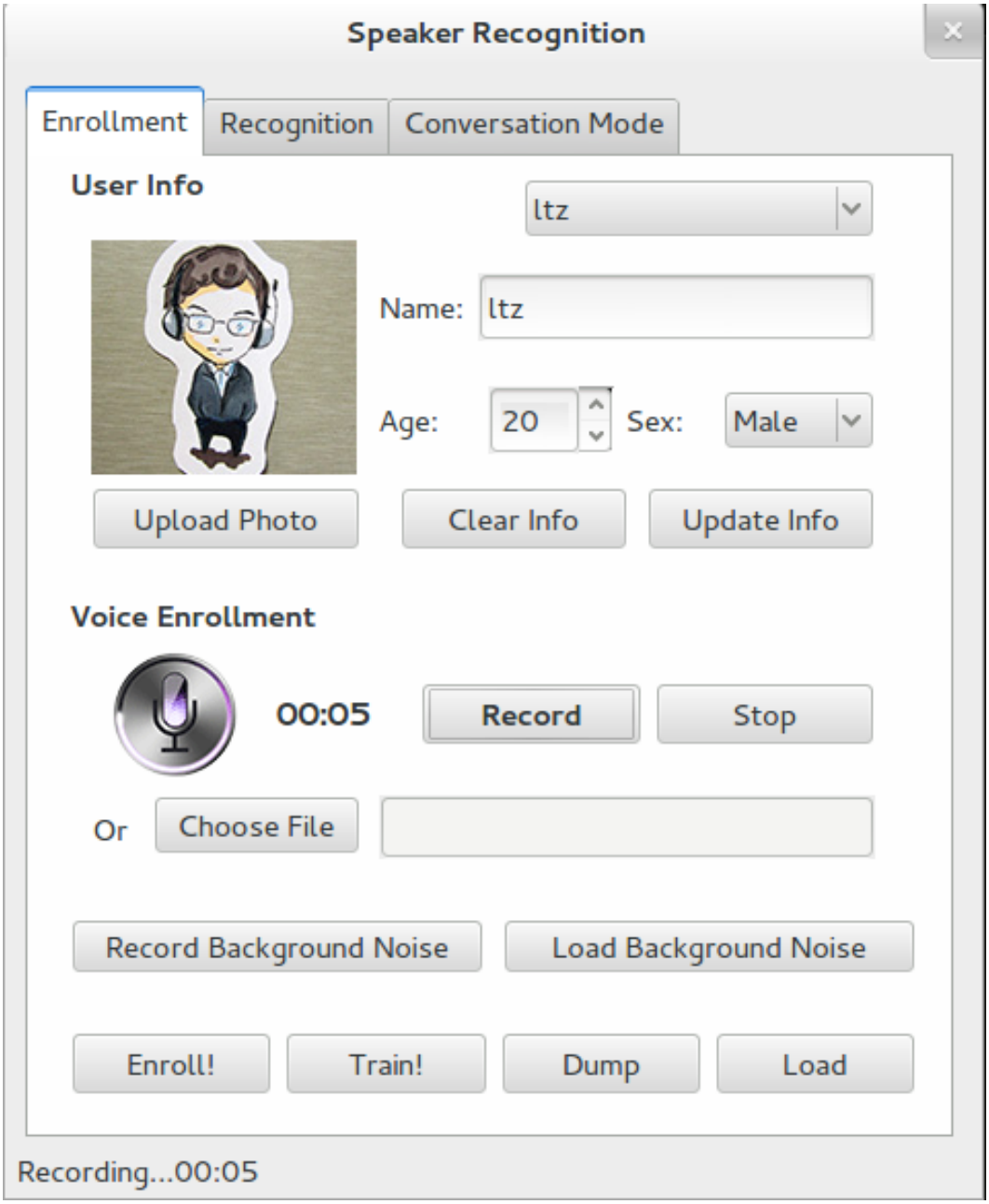
Figure 10

Result shown in Figure.10 indicates that, although CRBM have generic modeling ability, applying it on signal features does not fit our expectation. To achieve similar results, the training utterance should be twice as large as GMM used. Further investigation on using RBM to process signal features need to be conducted.

6 GUI

The GUI contains following tabs:

- **Enrollment**



A new user may start his or her first step by clicking the tab Enrollment. New users could provide personal information such as name, sex, and age. then upload personal avatar to build up their own data. Experienced users can choose from the userlist and update their infomation.

Next the user needs to provide a piece of utterance for the enrollment and training process.

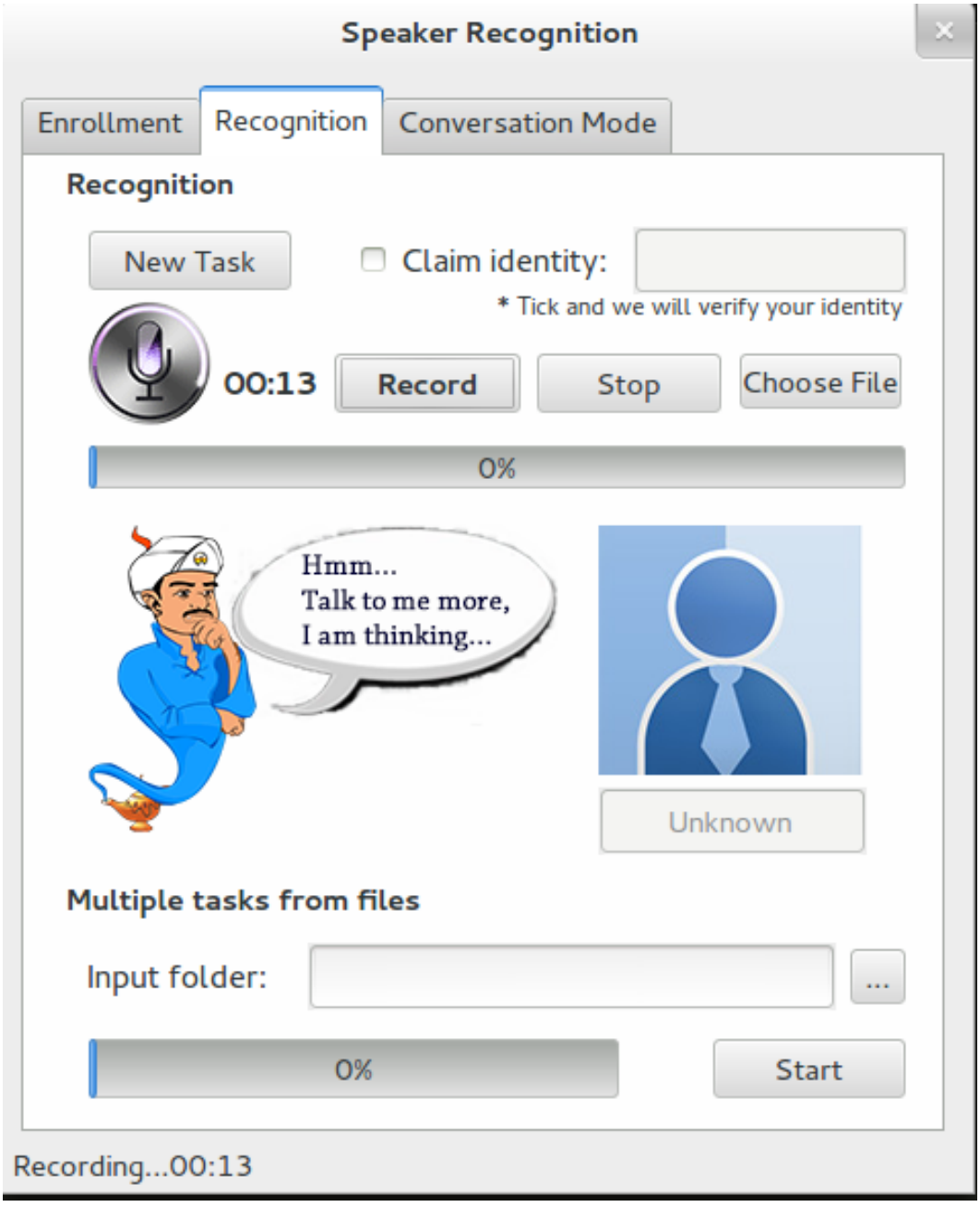
There are two ways to enroll a user:

– **Enroll by Recording** Click Record and start talking while click Stop to stop and save.There is no limit of the content of the utterance, whileit is highly recommended that the user speaks long enough to provide sufficient message for the enrollment.

– **Enroll from Wav Files** User can upload a pre-recorded voice of a speaker.(*.wav recommended) The systemaccepts the voice given and the enrollment of a speaker is done.

The user can train, dump or load his/her voice features after enrollment.

- **Recognition of a user**



An user present can record a piece of utterance, or provide a wav file, then the system will tell who the person is and show his/her avatar. Recognition of multiple pre-recorded files can be done as well, the result will be printed in the command line.

- **Conversation Recognition Mode**

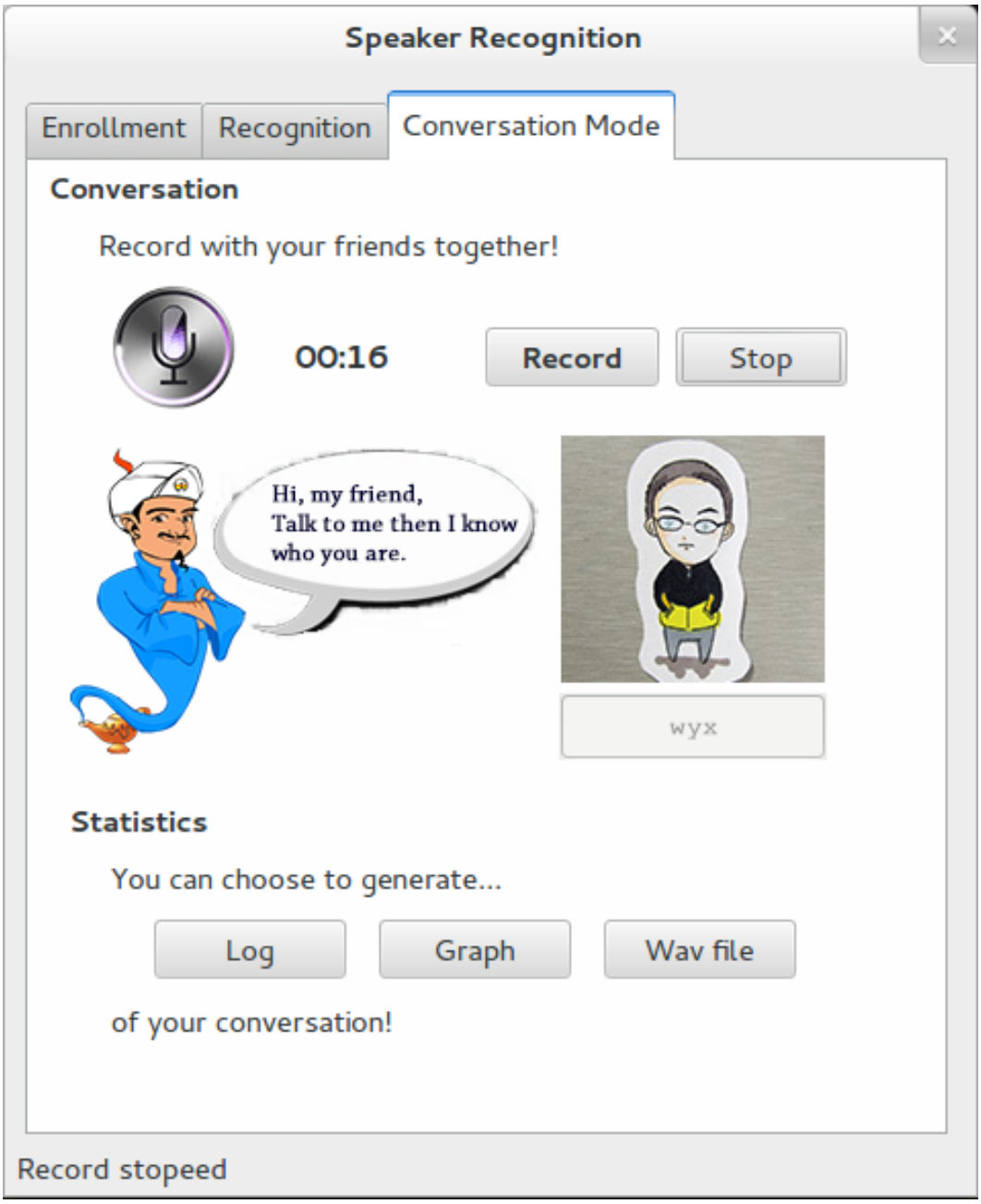
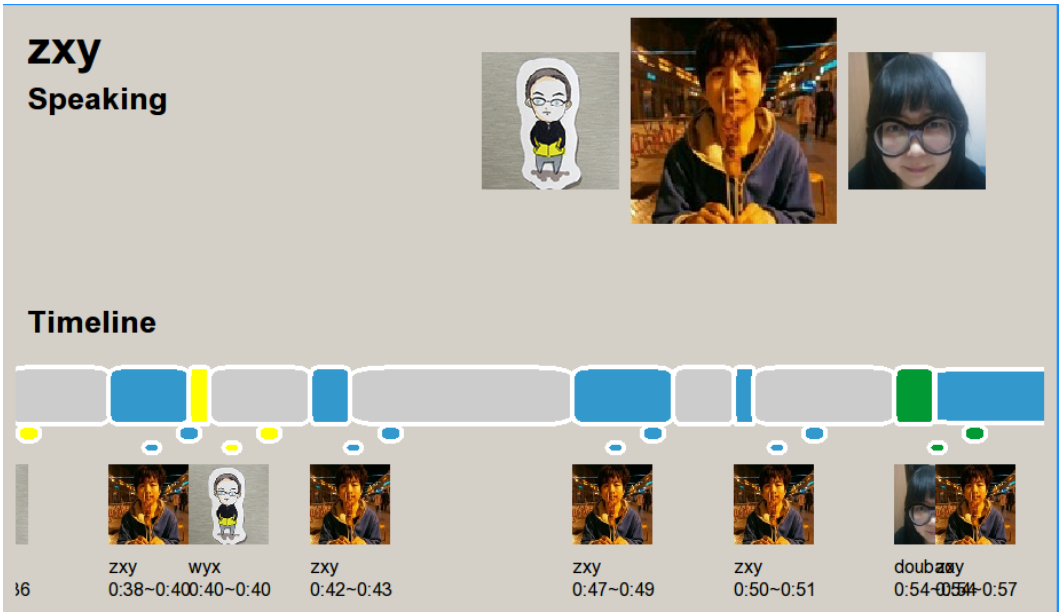


Figure 11

In Conversation Recognition mode, multiple users can have conversations together near the microphone. Same recording procedure as above. The system will continuously collect voice data, and determine who is speaking right now. Current speaker's anvatar will show up in screen; otherwise the name will be shown.

We can show a **Conversation flow graph** to visualize the recognition. A timeline of the conversation will be shown by a number of talking-clouds joining together, with start time, stop time and users' avatars

labeled. The avatar of the talking person will also be larger than the others. Different users are displayed with different colors in the timeline, and the timeline flows to the left dynamically just as time elapses.



7 References

[1] A. Anjos et al. "Bob: a free signal processing and machine learning toolbox for researchers". In: *20th ACM Conference on Multimedia Systems (ACMMM), Nara, Japan*. ACM Press, Oct. 2012. URL: http://publications.idiap.ch/downloads/papers/2012/Anjos_Bob_ACMMM12.pdf.

[2] David Arthur and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2007, pp. 1027–1035.

[3] Bahman Bahmani et al. "Scalable k-means++". In: *Proceedings of the VLDB Endowment* 5.7 (2012), pp. 622–633.

[4] Jeff A Bilmes et al. "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models". In: *International Computer Science Institute* 4.510 (1998), p. 126.

[5] Hsin Chen and Alan F Murray. "Continuous restricted Boltzmann machine with an implementable training algorithm". In: *Vision, Image and Signal Processing, IEE Proceedings-*. Vol. 150. 3. IET. 2003, pp. 153–158.

[6] George E Dahl et al. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 20.1 (2012), pp. 30–42.

[7] John Godfrey, David Graff, and Alvin Martin. "Public databases for speaker recognition and verification". In: *Automatic Speaker Recognition, Identification and Verification*. 1994.

[8] *Joint Factor Analysis Matlab Demo*. URL: http://speech.fit.vutbr.cz/software/joint-factor-analysis-matlab-demo.

[9] Patrick Kenny. "Joint factor analysis of speaker and session variability: Theory and algorithms". In: *CRIM, Montreal,(Report) CRIM-06/08-13* (2005).

[10] Patrick Kenny et al. "A study of interspeaker variability in speaker verification". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 16.5 (2008), pp. 980–988.

[11] Patrick Kenny et al. "Joint factor analysis versus eigenchannels in speaker recognition". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 15.4 (2007), pp. 1435–1447.

[12] *K-means clustering - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/wiki/K-means_clustering.

[13] *Levinson Recursion - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/wiki/Levinson_recursion.

[14] *LPC - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/wiki/Linear_predictive_coding.

[15] *MFCC - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/wiki/Mel-frequency_cepstrum.

[16] A-R Mohamed et al. "Deep belief networks using discriminative features for phone recognition". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 2011, pp. 5060–5063.

[17] *NumPy – Numpy*. URL: http://www.numpy.org/.

[18] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[19] *PyAudio: PortAudio v19 Python Bindings*. URL: http://people.csail.mit.edu/hubert/pyaudio/.

[20] *python speech signal processing library for education*. URL: https://pypi.python.org/pypi/pyssp.

[21] Javier Ramırez et al. "Efficient voice activity detection algorithms using long-term speech information". In: *Speech communication* 42.3 (2004), pp. 271–287.

[22] *Restricted Boltzmann machine - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/wiki/Restricted_Boltzmann_machine.

[23] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. "Speaker verification using adapted Gaussian mixture models". In: *Digital signal processing* 10.1 (2000), pp. 19–41.

[24] Douglas A Reynolds and Richard C Rose. "Robust text-independent speaker identification using Gaussian mixture speaker models". In: *Speech and Audio Processing, IEEE Transactions on* 3.1 (1995), pp. 72–83.

[25] *Scientific Computing Tools for Python*. URL: http://www.scipy.org/.

[26] *SoX - Sound eXchange*. URL: http://sox.sourceforge.net/.

[27] *Speaker Recognition - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/wiki/Speaker_recognition.

[28] *Talkbox, a set of python modules for speech/signal processing*. URL: http://scikits.appspot.com/talkbox.

[29] *The GPL licensed Python bindings for the Qt application framework*. URL: http://sourceforge.net/projects/pyqt/.

[30] *Universal Background Models*. URL: http://www.ll.mit.edu/mission/communications/ist/publications/0802_Reynolds_Biometrics_UBM.pdf.

[31] *Welcome to PyPR' s documentation! – PyPR v0.1rc3 documentation*. URL: http://pypr.sourceforge.net/.