Resource Usage Analysis for Speech Recognition Techniques

A
Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY
By

PULKIT VERMA
Under the Supervision of Dr. PRADIP K. DAS

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
May, 2015

Declaration

This is to certify that the thesis entitled "**Resource Usage Analysis for Speech Recognition Techniques**", submitted by me to the *Indian Institute of Technology Guwahati*, for the award of the degree of Master of Technology, is a bonafide work carried out by me under the supervision of Dr. Pradip K. Das. The content of this thesis, in full or in parts, have not been submitted to any other University or Institute for the award of any degree or diploma. I also wish to state that to the best of my knowledge and understanding nothing in this report amounts to plagiarism.

Signed: _____

Pulkit Verma
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati-781039, Assam, India.

Date: _____

Certificate

This is to certify that the thesis entitled "**Resource Usage Analysis for Speech Recognition Techniques**", submitted by Pulkit Verma (134101033), a master's student in the *Department of Computer Science and Engineering*, *Indian Institute of Technology Guwahati*, for the award of the degree of Master of Technology, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in my opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Signed: _____

Supervisor: Dr. Pradip K. Das
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati-781039, Assam, India.

Date: _____

Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor Dr. P. K. Das for giving me an opportunity to pursue this work under his guidance. His valuable guidance and encouragement at every step had provided a good basis for the thesis work. He was always available to discuss the problems faced all through the way.

I express my thanks to Indian Institute of Technology Guwahati and Department of Computer Science and Engineering for providing me the opportunity to take up this project and to do research work in this field.

I am also thankful to my family, Prof. Shivashankar B. Nair, Mr. Shashi Shekhar Jha, Mr. Tuhin Bhattacharya, Mr. Manoj Bode and Mr. Mayank Gupta for their support and words of encouragement. I would also like to thank my friends and classmates who have helped me throughout my stay at IIT Guwahati.

Last but not the least, I would like to express my heartfelt gratitude to the Government of India who funded this work, without which this thesis would not have been possible.

Sincerely
Pulkit Verma

Abstract

Speech recognition refers to the techniques that are used to analyze the speech signals so as to make sense out of them. A considerable amount of work has been done in this area and a lot of different approaches have been proposed. Despite all this, the size of speech data is still considered a major bottleneck in analysis of speech signals. One of the major drawbacks of large amounts of data is the performance impact of speech processing applications on low resource devices like mobile phones, tablets, etc.

Resource usage is an important factor into consideration while developing speaker recognition applications for mobile devices. Sometimes usage parameters are considered as important as accuracy of such systems. In this work, an analysis of resource utilization in terms of power consumption, memory and space requirements of three standard speaker recognition techniques, viz. GMM-UBM framework, Joint Factor Analysis and i-vectors is presented.

The experiments are performed on the MIT MDSVC corpus using Energy Measurement Library (EML). It is found that though i-vector approach requires more storage space, it is better than the other two in terms of memory and power consumption, the two important factors for software performance in present times.

Keywords: GMM, UBM, JFA, i-vectors, speaker recognition, resource usage, power consumption

Contents

List of Figures

List of Tables

Chapter 1
Introduction

SPEECH recognition is the process of identifying the text equivalent of spoken sentence through a machine. A major issue in speech recognition by a machine is no two speech utterances are identical even if the spoken utterances are same. In other words we may say that speech data (speech samples) is highly variable, random and ambiguous in nature.

The work on recognizing speech has been in constant progress since it was first envisioned. A number of methods have been proposed and implemented for the same. But still any such system requires a large amount of training data to function with acceptable accuracy. This consumes a lot of resources. Also the real time recognition of speech is many times not possible pertaining to various resource constraints.

Increasing number of people now-a-days possess mobile phones with capabilities of supporting speech recognition applications. The real challenge however is to make the speech applications less resource hungry so that they can be practically used.

Memory and secondary storage space has been the standard parameters taken into consideration whenever resource usage of any software is considered. But in the current scenario with number of mobile devices working on temporary power sources like batteries increasing in exponential numbers, power consumption of devices is becoming an equally important factor for considering the performance of applications.

This thesis explores the resource utilization in terms of power, memory and secondary storage space requirements of the state of the art speech recognition techniques. This

exploration is limited to the probabilistic model based techniques like the GMM-UBM approach, Joint Factor Analysis and Front End Factor Analysis.

We first find the best configuration of each method by changing the scoring and normalization method. Then once the best performing combination of each technique is obtained they are all compared with each other.

The rest of this thesis is organized as follows: Chapter 2 will review some of the basic concepts related to speech recognition, after which Chapter 3 will introduce the three methods which are to be compared with each other. The concept of measuring the resource usage will be introduced in Chapter 4. Chapter 5 will explain the experimentation process and analysis of results obtained will be done. Finally, Chapter 6 concludes this thesis with summary and possible works that can be taken up in future.

Chapter 2
Background

This chapter layouts the groundwork for understanding the work presented in this thesis. After a brief introduction to the related works that have been proposed over time, a short explanation of typical speech recognition systems will be given.

2.1 Literature Survey

Gaussian Mixture Model (GMM) based speaker verification approach was proposed by Reynolds [28] a long time back, it remained very popular in the form of GMM-UBM framework. It was modified so as to take into account the inter-speaker (speaker) variability and intra-speaker (channel) variability in the modeling process. This approach was termed as Joint Factor Analysis (JFA) [18, 19].

JFA considers that these variabilities are independent of each other. But it was later found out that channel variabilities were also modeling the inter-speaker variations [9]. Dehak et. al. [11] resolved this issue by proposing a channel-blind approach where speaker and channel variabilities are modeled together in a low dimensional total variability space. The vectors represented in this space are called i-vectors. This has become the latest state of the art approach because of its low complexity and better performance.

Calculating the power consumption of a software was first proposed for embedded software [31]. Making such systems useful for application software is very tricky and many methods have been developed but very few of them are actually useful. A review of

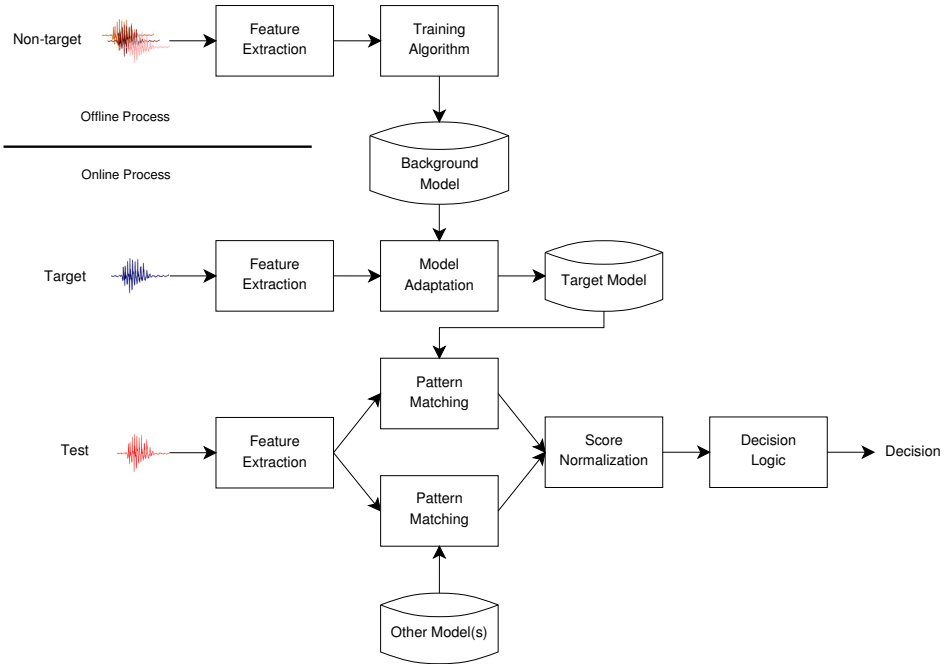


FIGURE 2.1: Flowchart for a generic Speaker Recognition Process

energy consumption of software libraries was given in [24] and [25]. PowerAPI [2] and Energy Measurement Library (EML) [8] are some of the methods which successful implementations solving this problem.

In this thesis, a comparative study of power, memory and storage space consumption of GMM-UBM framework, JFA and i-vector approach in the domain of speaker recognition is presented.

2.2 Generic Approach

The general framework of any speech recognition application can be explained using a speaker verification system as shown in Figure 2.1. Applying it to any other domain requires changing the suitable parameters while giving speech data as input to the system. This generic approach can be subdivided into two main steps, speaker enrollment and speaker verification.

During the speaker enrollment process, a background model is generated using the data collected from non-target utterances. In this paper, for nearly all the approaches this model will be Universal Background Model (UBM). Using this background model, a target model is generated using the target utterances by adapting the background model according to the target data. Now this target model will act as the single point of reference for the pattern recognition algorithms. Whenever any test utterance is given

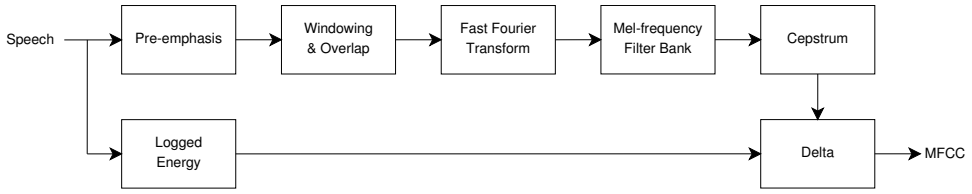


FIGURE 2.2: Flowchart for MFCC extraction [12]

as input to the system, features are extracted from it and pattern matching algorithms are applied on it using one or more kinds of target models. The resultant similarity score is then normalized and final decision is taken after applying some decision logic to this normalized score.

2.3 Feature Extraction

An important aspect of using any of the techniques used in speech recognition applications is the feature extraction step. The features used for this purpose must fulfill some criteria so as to give better performance. According to Wolf [35], some of the important characteristics of these feature vectors are that they should:

- be less susceptible to environmental noise
- occur frequently and naturally in normal speech
- not be affected by health of the speaker
- be easily measurable
- be difficult to copy by impostors

Due to these reasons generally MFCCs (Mel-frequency cepstral coefficients) [4] are used as feature vectors for speech recognition applications. A simple flowchart depicting the MFCC extraction process is depicted in Figure 2.2.

Chapter 3
Factor Analysis Techniques

This chapter outlines the factor analysis techniques used in speech recognition applications. The main concept behind this approach is to project the higher dimensional feature vectors on a low dimensional space where it is represented by factors. Each factor analysis method differs from each other depending on which combination of features is projected on lower dimensional space.

This chapter will also explain the basic three methods compared for resource consumption in this work.

3.1 The GMM-UBM Approach

This is one of the most commonly used approach for speech recognition tasks. The framework is similar to the generic speaker recognition method explained earlier. Here Estimation Maximization (EM) algorithm is used as the training algorithm and Universal Background Model (UBM) acts as the background model. Maximum A Posteriori (MAP) is used to adapt the UBM to generate the target models. Figure 3.1 gives the flowchart for the working of this framework.

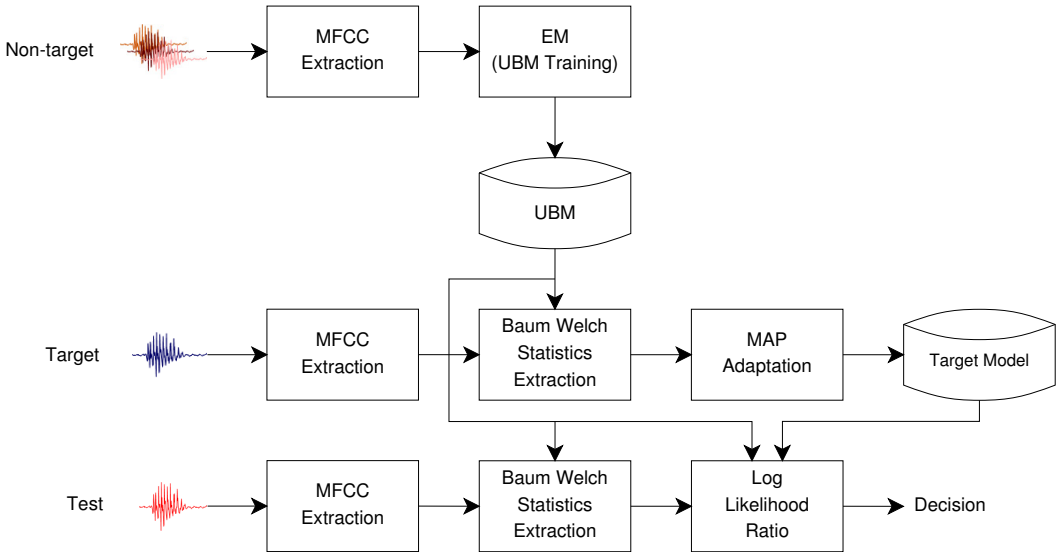


FIGURE 3.1: Flowchart for GMM-UBM Approach

3.1.1 Gaussiam Mixture Model (GMM)

A Gaussian mixture model [26] is a weighted sum of M component Gaussian densities as given by the equation,

$$p(x|\lambda) = \sum_{i=1}^{M} w_i \; g(x|\mu_i,\Sigma_i) \tag{3.1}$$

where,

x is a D-dimensional continuous-valued data vector (i.e. measurement or features);
w_i, i =1, ..., M, are the mixture weights; and
$g(x|\mu_i,\Sigma_i)$, $i = 1, ..., M$, are the component Gaussian densities.

It should be noted that the mixture weights are such that $\sum_i w_i = 1$ and each component densities is a D-variate Gaussian function of the form,

$$g(x|\mu_i,\Sigma_i) = \frac{1}{(2\pi)^{\frac{F}{2}}|\Sigma_c|^{\frac{1}{2}}} \exp\Big\{-\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i)\Big\} \tag{3.2}$$

with mean vector μ_i and covariance matrix Σ_i.

The complete Gaussian mixture model is parameterized by three parameters:

- Mean vectors,
- Covariance matrices, and
- Mixture weights from all component densities.

These parameters are collectively represented by the notation,

$$\lambda = \{w_i, \mu_i, \Sigma_i\}, i = 1, ..., M \tag{3.3}$$

If training vectors/data and GMM configuration is known, next step is to estimate λ, such that it resembles the distribution of training features. Various techniques to achieve this are already present [23]. The most popular method is maximum likelihood (ML) estimation. The estimates of ML parameters are calculated using expectation-maximization (EM) algorithm [13].

GMMs are very popular in Speech Processing Systems since they can model a large class of speech characteristics. Even for arbitrary component densities the model gives a good approximation. The model is also capable of modeling the hidden characteristics of speech data generated by various phonetic entities like vocal cords, voice tract, etc.

During implementation, GMMs are used to grab the distribution of MFCCs in the feature space. It is assumed that given these feature vectors $u = y_1, y_2, y_3, ...y_L$ are independent of each other. Hence the probability of a given utterance u given the model λ is simply the product of each L frames. Since L could be large and probabilities are less than or equal to 1, such a large product may lead to underflow. So log likelihood is used in the computations as follows:

$$log\ p(u|\lambda) = \sum_{i=1}^{L} log\ p(y_t, \lambda) \tag{3.4}$$

3.1.2 Estimation Maximization

The Estimation Maximization (EM) algorithm can be used to estimate the maximum likelihood (ML) parameters of a model λ for a set of training vectors. The EM algorithm refines the parameters of GMM iteratively. The likelihood of the estimated model for the observed feature values increases by this refinement.

3.1.3 Universal Background Model (UBM)

Universal Background Model (UBM) [27] represents a general speaker independent model which is actually a UBM model trained on very large amount of speech data. It models the speaker independent feature distribution. Hence this model is expected to contain the features of any utterance on which system is tested.

To estimate the GMM parameters (λ), Maximum A Posteriori (MAP) Parameter Estimation algorithm [1] is generally used. This estimation algorithm is generally used when

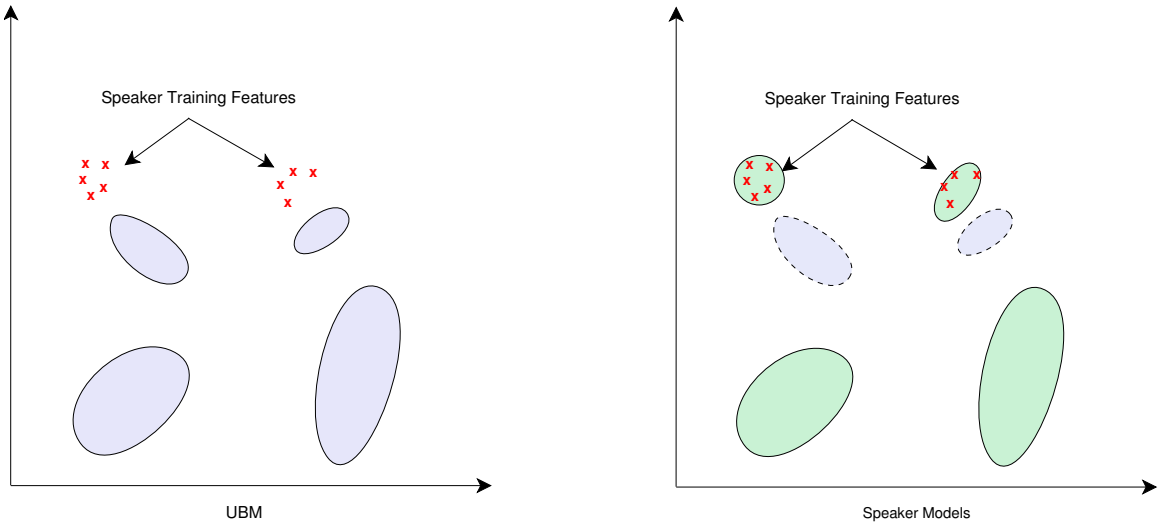


FIGURE 3.2: Example of MAP adaptation [29]

a general model is adapted using very limited training data. Still the choice of method to estimate the parameters varies from implementation to implementation.

3.1.4 MAP Adaptation

The next logical step is to generate the models for all the classes for which the recognition task is to take place. Hence in speaker recognition task the aim is to generate models for each speaker. EM algorithm cannot be applied here as data for each speaker is expected to be small and may not be representative of that speaker's speech features. Also by using EM channel characteristics might be modeled within speaker models.

The solution to this problem is to use UBM itself to generate the speaker models. Hence the well trained parameters from UBM are updated using maximum a posteriori (MAP) adaptation. The implementation of MAP is similar to EM, but Baum Welch statistics [Appendix A] are used here to updated the UBM. Figure 3.2 depicts an exmple where MAP is used to adapt UBM based on speaker features to generate the speaker models.

3.2 Joint Factor Analysis

3.2.1 Methodology

In Joint Factor Analysis [19–21], a supervector M is used to represent any speech utterance. M contains speaker and channel dependent supervectors.

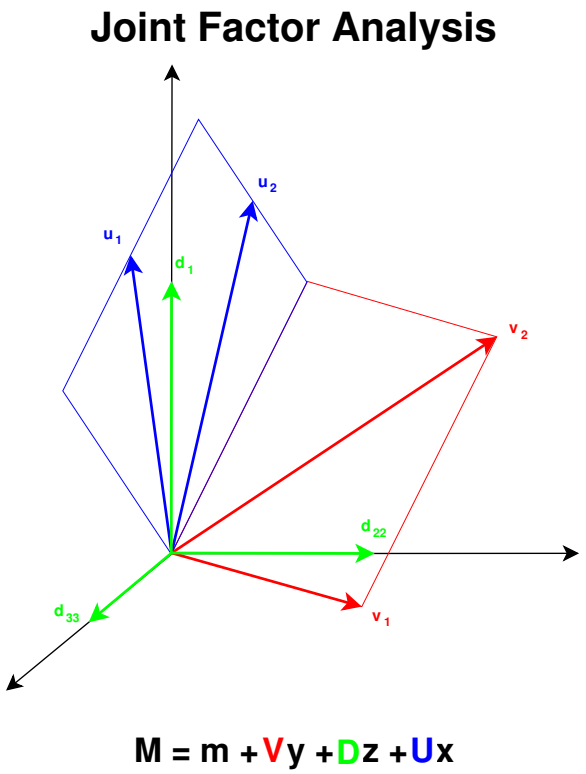


FIGURE 3.3: Possible projections in Joint Factor Analysis [7]

Let C be the number of components in UBM and F be the dimension of acoustic feature vectors.

Now for a given utterance, we concatenate the F-dimensional GMM mean vectors to get the supervectors of dimension CF. For a particular speaker, we assume that the speaker and channel dependent supervector M is generated by the vector sum of speaker dependent supervector and channel dependent supervector. Also these speaker and channel supervectors are distributed normally and are statistically independent.

$$\mathrm{M} = \mathrm{s} + \mathrm{c} \tag{3.5}$$

where,

M is speaker and channel dependent supervector;

s is speaker dependent supervector;

c is channel dependent supervector.

Both the speaker and channel factors are decomposed into low dimensional set of factors. Each of these low dimensional factors operate along the principal dimensions of the corresponding component.

We assume that the distribution of speaker dependent supervector s has a hidden variable description of the form:

$$s = m + Vy + Dz \tag{3.6}$$

where,

s is speaker dependent supervector;

m is CF $\times$ 1 speaker and channel independent supervector (from UBM);

V is eigenvoice matrix (rectangular matrix of low rank);

D is CF $\times$ CF residual diagonal matrix;

y is a vector representing speaker factors;

z is a normally distributed CF dimensional random vector representing speaker specific residual factors.

The decomposition of a factor into lower dimensional factors can be illustrated by the following decomposition of speaker dependent component of speaker factor:

$$V * y = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \dots & v_N \\ | & | & & | \end{bmatrix} * \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{bmatrix} \tag{3.7}$$

where,

V is eigenvoice matrix;

y is lower dimensional vector representing speaker (or eigenvoice) factors.

Each speaker factor y_i controls corresponding v_i.

We assume that the distribution of channel dependent supervector c has a hidden variable description of the form:

$$c = Ux \tag{3.8}$$

where,

c is channel dependent supervector;

U is eigenchannel matrix (matrix of low rank);

x is a normally distributed random vector representing channel factors.

With each mixture component i, a diagonal covariance matrix Σ_i is associated. The variability in acoustic observation vectors is not modeled by either (3.5) or (3.8).

We now define a CF $\times$ CF super-covariance matrix Σ whose diagonal is obtained by concatenating all the covariance matrix Σ_i. The variability that is not modeled by s and c is modeled by Σ.

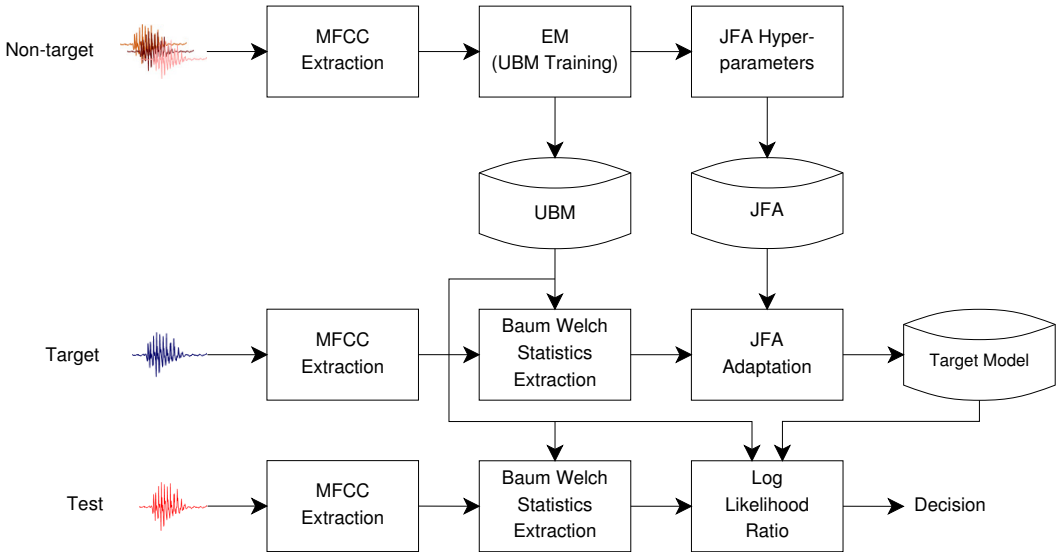


FIGURE 3.4: Flowchart for Joint Factor Analysis [12]

3.2.2 Working

Once the MFCCs for all utterances in the training set are extracted, Estimation Maximization is applied to generate the Universal Background Model (UBM). JFA hyper-parameters are also extracted as explained in Kenny [18]. For the Target dataset, once the MFCCs are extracted, zeroth and first order Baum-Welch statistics are extracted. Now using the JFA hyper-parameters extracted earlier adaptation is done on the data to generate the target model.

For performing the recognition of any test utterance, MFCCs are extracted followed by computation of the Baum-Welch statistics. Now using the target model and these statistics values the log likelihood is calculated for each speaker. The maximum value is chosen as the recognized speaker. Figure 3.4 explains this process in detail.

It should be noted that log likelihood calculation is not the only scoring method used with JFA and its choice may vary across applications. Various such scoring methods used with JFA for speaker recognition are compared in Glembek et al. [15]. It is deduced that though the performance of the various techniques does not vary significantly, the real difference comes in the speed of scoring. Linear scoring is found to be fastest of all the scoring techniques as the channel compensation is calculated only once per speech utterance.

The various algorithms and detailed description of Joint Factor Analysis can be found in Kenny [18].

3.3 Front End Factor Analysis

In JFA it is assumed that the channel factors will only model the channel effects, but it was observed by Dehak [9] that the channel dependent supervector also models the speaker features. To take this finding into account, a new approach was proposed where there was no distinction between channel and speaker variabilities.

A new low dimensional total variability space T was introduced to account for both the variabilities, where M is given by:

$$M = m + Tx \tag{3.9}$$

where,

m is the UBM supervector (speaker and channel independent supervector);
x is a normally distributed random vector in this space; and
T is a low ranked rectangular matrix.

The factors of x are also called as total factors. The new vectors are known as identity vectors or i-vectors.

In this approach, it is assumed that M is distributed normally with TT' as its covariance matrix and m as its mean vector. Here total variability matrix T is obtained in the same manner as that of V, but it is assumed that the utterances of the same speaker are produced by different speakers unlike in the process of training eigenvoice matrix V.

In order to bring down computation, the channel compensation is done in total factor space as the dimensionality of i-vectors is lower as compared to GMM supervectors. Hence the computation cost is much less as compared to JFA.

One of the additional advantage of this approach is that supervised training is not needed in this model unlike JFA and GMM-UBM.

3.4 Parameter Training

While training the total variability matrix T, it is assumed that each utterance of the training set belongs to a different speaker, unlike JFA where all recordings from a same person are assumed to be generated from the same speaker.

The posterior distribution of w can be determined using Baum Welch Statistics [Appendix A] from the UBM [11].

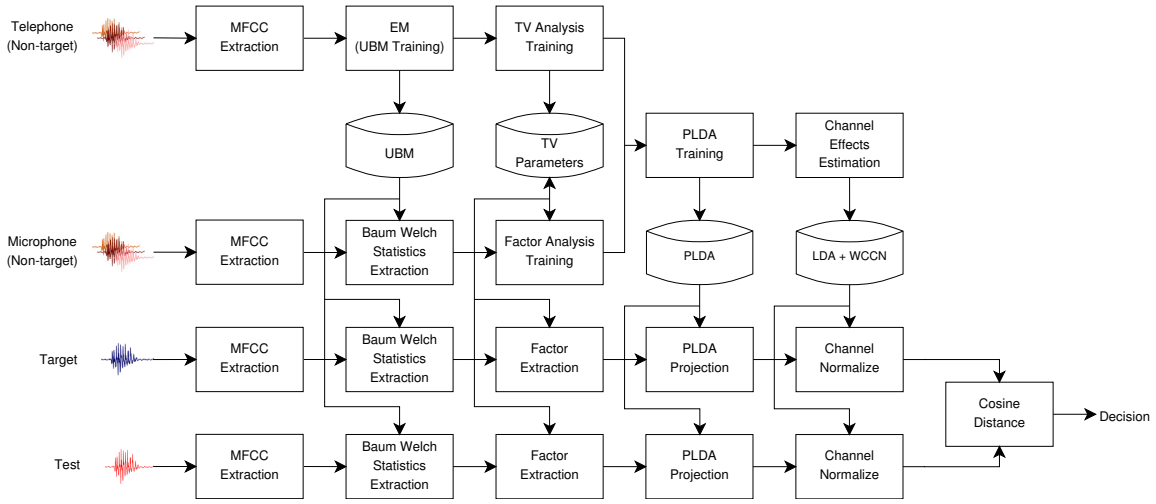


FIGURE 3.5: Flowchart for channel blind i-vector based approach [12]

3.5 Channel Blind Approach

Most of the speaker recognition systems are trained on either telephone speech or microphone speech. Dehak et al. [10] proposed a channel-blind system which is conditioned on both telephone and microphone speech. To project both types of data on same space, PLDA was used. And for the hyper-parameter training, first μ, V and Σ were trained on telephone data assuming U = 0 (EM algorithm was used). Then U was trained on microphone data assuming μ, V and Σ are fixed (Maximum a posteriori probability (MAP) estimate was used). This approach is explained in detail in Figure 3.5. This is the general template for nearly all the approaches that involve i-vectors. In the literature, there are reported works [1, 14] where efficient computations of i-vectors are presented. There are several other variations of this general approach which are listed below:

1. Using different features instead of MFCCs.
2. Using different feature or score normalization technique instead of PLDA, LDA and WCCN.
3. Modifying the statistics calculation.
4. Using different scoring technique instead of Cosine Distance scoring.
5. Changing some inherent assumption of Front End Factor Analysis.

All these methods try to alter the standard approach so as to improve certain aspects of the recognition system.

3.6 Discussion

In this chapter, a concise overview of various speech recognition methods that uses probabilistic models was presented. The recognition performance of these methods vary across the domains. Some of them are highly domain or problem specific and generally perform well in ideal laboratory environments. The main challenges are found to be the paucity of training data, choice of training data, length of utterances, background/environmental noise, emotional state of speaker, etc. Hence most of the methods fail when tested in real life environments. Though some of the methods looks highly promising, much work needs to be done in coming up with a generic method that is suitable to tackle all the challenges mentioned. Another important criteria for use of i-vectors is the execution time of the recognition process. Though some methods might give very good performance, if the running time on real data is high, they may render the application useless for most practical situations.

Chapter 4
Resource Utilization Measurement

This chapter outlines the resource measurement strategies used in this work. The resource measurement is done in terms of three parameters, viz, power, memory and secondary space. These parameters give a good outline of the application performance in terms of its usability. This becomes more important in current times when mobile devices users have increased in exponential numbers.

4.1 Power Measurement

We have used Energy Measurement Library (EML) [8] to measure the power requirements of the various approaches. It acts as a middleware between code for measuring power consumption and hardware based measurement tools. Single multi-core Intel® Xeon® CPU was used for the experiments. EML also provides abstracted interface to simplify data collection and representation. It is currently available for a limited number of hardware.

4.1.1 Energy Measurement Library

EML provides easy measurement of energy consumption of software systems. The options to speed up the measurement are also available. It abstracts the measurement process from the software hence relieving the developers to take care of measurements on various architectures and remember to avoid clashes with the interfaces for measurement.

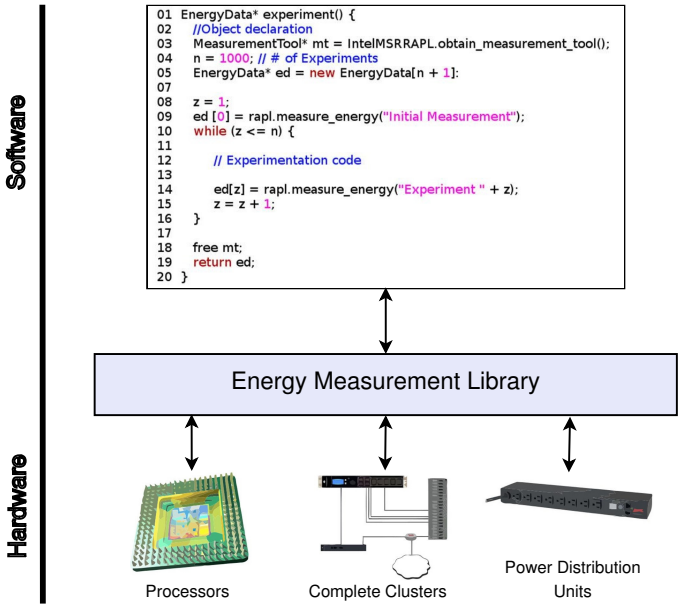


FIGURE 4.1: EML structure diagram overview [8]

EML uses Performance API (PAPI) [5] to get the power readings via RAPL register interfaces directly.

The programming interface provided to the source codes for instruments is given in an unified form. A configuration file is needed to initialize the measurement tool. When the object for measurement tool m is added a timestamp $t(m)_{init}$ is saved. Hence after the measurement calls when the results are returned the corresponding timestamp $t(m)_{curr}$ is also returned. This data along with other metadata can be configured to be saved in JSON format.

The power measurement support provided by EML is for three kinds of devices currently:

1. Intel CPUs:
   (a) Intel CPUs Sandy Bridge and later: Data from these devices is collected via Intel RAPL MSR (Model Specific Registers) interface for the Intel Xeon family.
   (b) Intel Xeon Phi: Data from these device is read via Intel MPSS 3.x (Many Platform Software Stack)
2. Nvidia Fermi and Kepler cards: Data is read via NVML (Nvidia Measurement Library).
3. Metered Power Distribution Units (PDU): The support is currently only for a standard APC metered PDU and a custom Schleifenbauer PDU. The data collection is also limited for this as data can only be collected at intervals of about 2-3 seconds.

4.2 Memory Measurement

For the memory requirements of the applications, we have calculated the resident set size (RSS) portion of the memory as it is the actual amount of memory that is occupied by the process in main memory. We have used `pmap` utility available in UNIX based operating systems for this purpose.

4.2.1 `pmap` Utility

`pmap` gives the following memory information related to each map of memory map of a process:

1. Start address of the map.
2. Map size in kilobytes.
3. Resident Set Size in kilobytes.
4. Dirty pages (both shared and private) in kilobytes.
5. Read, write, execute, shared and private (copy on write) permission on map
6. Mapping type. It can be either (1) a file name; (2) `[ anon ]` for allocated memory; or (3) `[ stack ]` for program stack.
7. Offset of map into the file
8. Device name

Since we only require the resident set size for the measurement purposes, we extract that value from all the maps and sum them to get the total RSS at that instant. This total RSS value is calculated at fixed intervals.

4.3 Space Measurement

For the space requirements we have calculated the amount of space occupied by the executables on the secondary disk. `ls` command was used for this purpose.

Chapter 5
Comparing Resource Usage

This chapter outlines the experiments performed to calculate and compare the resource usage of the various speaker recognition strategies. The platform specification and the data/corpus on which the experiments are performed are also given.

5.1 Corpus

In this work, the MIT Mobile Device Speaker Verification Corpus (MDSVC) [36] was used which is aimed at supporting speaker verification research using mobile devices. The recording was done using microphones and internal headsets in different environments to generate multi-style trained models. Since the data was collected over 2 sessions it contains inter-session variability as well.

The corpus has enrolled data of 48 speakers out of whom 26 were male and 22 were female. A total of 54 recordings per person were carried in each session. Hence over 2 sessions, 5,184 recordings were collected for enrolled users. Imposter data was also recorded which consisted of 40 speakers, out of which 23 were male and 17 female.

The text being read consisted of names and ice cream flavor phrases, hence the average length of each recording is about 2 seconds which makes the training and recognition task a bit difficult. Since the amount of data is also quite less, it adds to the difficulty of recognition.

5.2 Platform

The output of power requirements vary across systems and might change drastically if the measurement platform is changed. But the relative power requirements of different process generally does not change.

The system used for the experiments has the following specifications:

- 1 x Intel® Xeon® CPU E5530 @ 2.40GHz
- 16 Cores
- 48 MB L3 Cache
- 32 GB RAM
- gcc 4.1.2
- Intel® MSR RAPL interface
- Intel® MPSS 3.4.2

5.3 Toolkit

For the speaker recognition techniques, program development was based on ALIZE 3.0 [22] toolkit. This was to ensure that standard code-base is used for the experiments. The configuration parameters that can be set for using this toolkit can be found in Appendix B.

5.4 Gaussian Mixture Models

Three different scoring methods have been used with Gaussian Mixture Models and the resource usage is calculated and compared. The three scoring methods are:

1. Z-norm: Z-norm or Zero normalization compensates for inter-speaker score variation. The normalization statistics for it are calculated offline. It allows to use a global speaker independent decision threshold. Normalization score for Z-norm is given by,

$$LLR(Y_{test}, S)_{norm} = \frac{LLR(Y_{test}, S) - \mu(S)}{\sigma(s)} \tag{5.1}$$

where,

LLR is log likelihood ratio;

Y_{test} is the test utterance;

S is the claimed speaker; and

μ, σ are the normalization statistics.

For estimating the normalization statistics, a set of impostor trials are scored against the target model and the mean and the standard deviation of the scores are computed.

2. T-norm: T-norm or Test normalization compensates for inter-session score validation.The normalization statistics for it are calculated online. It attempts to reduce the overlap between impostor and true score distributions of each client-speaker. Normalization score for T-norm is given by,

$$LLR(Y_{test}, S)_{norm} = \frac{LLR(Y_{test}, S) - \mu(Y_{test})}{\sigma(Y_{test})} \tag{5.2}$$

where,

LLR is log likelihood ratio;

Y_{test} is the test utterance;

S is the claimed speaker; and

μ, σ are the normalization statistics.

3. ZT-norm: Zero-dependent test-score normalization (ZT-norm) [37] has been used in GMM-UBM approach and JFA approach. In this technique T-norm of each test utterance is calculated against each impostor model. These scores are further normalized by applying Z-norm to get the output of ZT-norm. Here it should be noted that Z-norm are calculated by scoring each target speaker model with each impostor utterance.

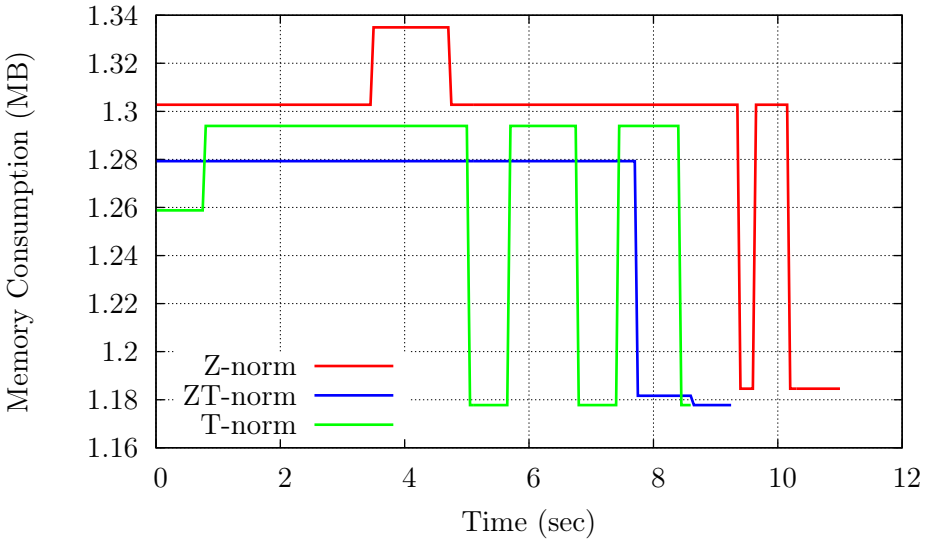
The graph in Figure 5.1a shows the variation of memory usage with time for the various scoring techniques used with GMM. It can be clearly seen that ZT-norm takes less time compared to the other two methods to complete its execution. Also its memory map is much smaller in size, which makes it the best suitable normalization and scoring technique out of the ones which were tested. Figure 5.1b depicts the comparison of power and space in graphical form.

5.5 Joint Factor Analysis

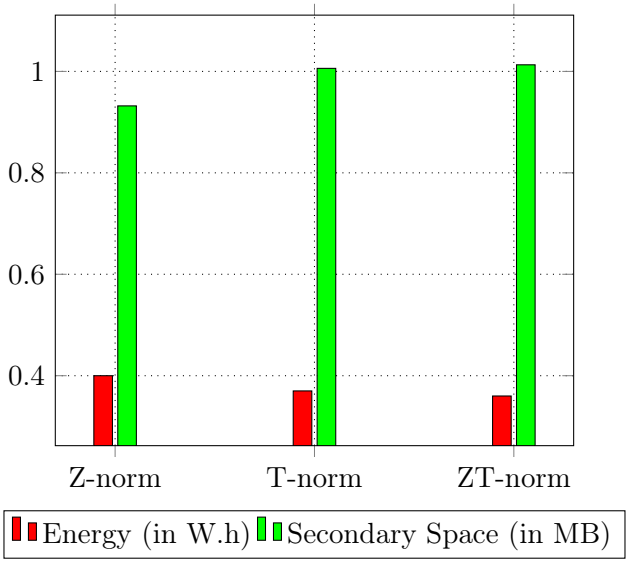
Joint Factor Analysis was used with dot product metric to normalize the scores. Due to the complexity of the JFA, only one scoring method was used with JFA. Dot product

TABLE 5.1: Resource consumption for different scoring methods when used with GMM-UBM approach for speaker recognition

Approach	**Memory Consumption (MB)**		**Energy**	**Space**	**Time**
	Average	**Maximum**	**(W.h)**	**(KB)**	**(sec)**
Z-norm	1325.7	1367	0.40	932	11.05
T-norm	1294.4	1325	0.37	1006	9.30
ZT-norm	1299.6	1310	0.36	1013	8.65



(A) Memory Consumption



(B) Energy and Space Requirements

FIGURE 5.1: Resource consumption for different scoring methods when used with GMM-UBM approach for speaker recognition

TABLE 5.2: Resource consumption for dot scoring method when used with Joint Factor Analysis for speaker recognition

Approach	**Memory Consumption (MB)**		**Energy**	**Space**	**Time**
	Average	**Maximum**	**(W.h)**	**(KB)**	**(sec)**
Dot Scoring	1292	1488	0.36	1034	6.45

metric is given by,

$$Score(Y_{test}, S) = Y_{test}^{'}.S \tag{5.3}$$

where,

Y_{test} is the test utterance; and

S is the claimed speaker.

Figures 5.2a and 5.2b depicts the resource consumption for Joint Factor Analysis when used with dot product scoring technique. Table 5.2 also depicts the same data.

5.6 i-vectors

Four different scoring methods have been used with i-vectors and the resource usage is calculated and compared. The three scoring methods are:

1. MD + EFR: It is Mahanalobis Distance scoring along with Eigen Factor Radial (EFR) [3] normalization. Mahanalobis based scoring function for speaker detection is given as,

$$Score(Y_{test}, S) = (Y_{test} - S)^{'} W^{-1} (Y_{test} - S) \tag{5.4}$$

where,

Y_{test} is the test utterance;

S is the claimed speaker; and

W is the covariance matrix computed on the EFR normalized i-vectors.

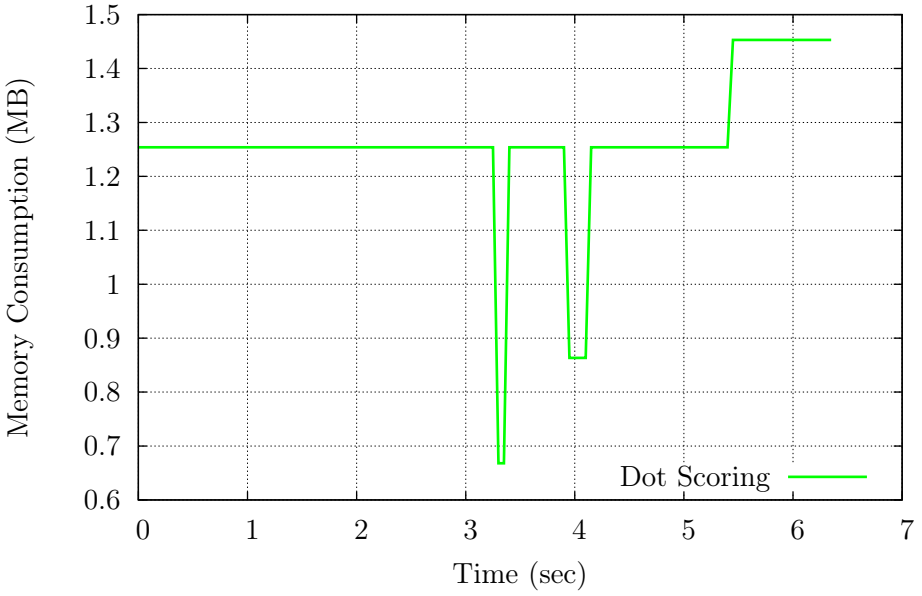
2. CS + WCCN: It is Cosine Similarity when used with Within Class Covariance Normalization (WCCN). Cosine similarity for speaker recognition is given as,

$$Score(Y_{test}, S) = \frac{(Y_{test})^{'} S}{||\mathrm{Y_{test}}||.||\mathrm{S}||} \tag{5.5}$$

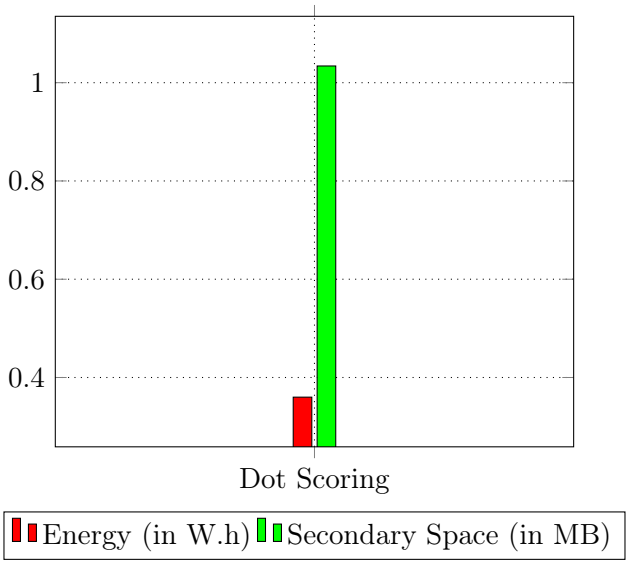
where,

Y_{test} is the test utterance; and

S is the claimed speaker.



(A) Memory Consumption



(B) Energy and Space Requirements

FIGURE 5.2: Resource consumption for dot scoring method when used with Joint Factor Analysis for speaker recognition

3. PLDA: Probabilistic Linear Discriminant Analysis (PLDA) [17] has been used for similarity scoring in the i-vector approach. Given two i-vectors x_1 and x_2, the PLDA score measures the log-likelihood ratio between the two hypotheses $\{H_0, H_1\}$, where H_0 hypotheses that x_1 and x_2 belong to the same speaker while H_1 hypotheses otherwise. The score is given as:

$$score = log(p(x_1, x_2|H_0)) - log(p(x_1, x_2|H_1)) \tag{5.6}$$

This can be explained by the fact that if the i-vectors belong to the same speaker then their latent variables will be same, otherwise they will be different. The solution to (5.6) can be found in [17]. For normalization standard Length Normalization is used prior to applying PLDA.

4. 2-Covariance Scoring: The two-covariance model can be seen as particular case of PLDA. Let an i-vector w of a speaker s be decomposed as,

$$w = y_s + \epsilon \tag{5.7}$$

where, y_s is the speaker model of same dimension as w, ϵ is the Gaussian noise, and

$$P(y_s) = \mathcal{N}(\mu, B) \tag{5.8}$$

$$P(w|y_s) = \mathcal{N}(y_s, W) \tag{5.9}$$

where,

$\mathcal{N}$ represents normal distribution;
μ is training dataset mean;
B is between-speaker covariance matrix; and
B is within-speaker covariance matrix.

Now according to assumptions 5.8 and 5.9, the score given in 5.6 can be redefined as,

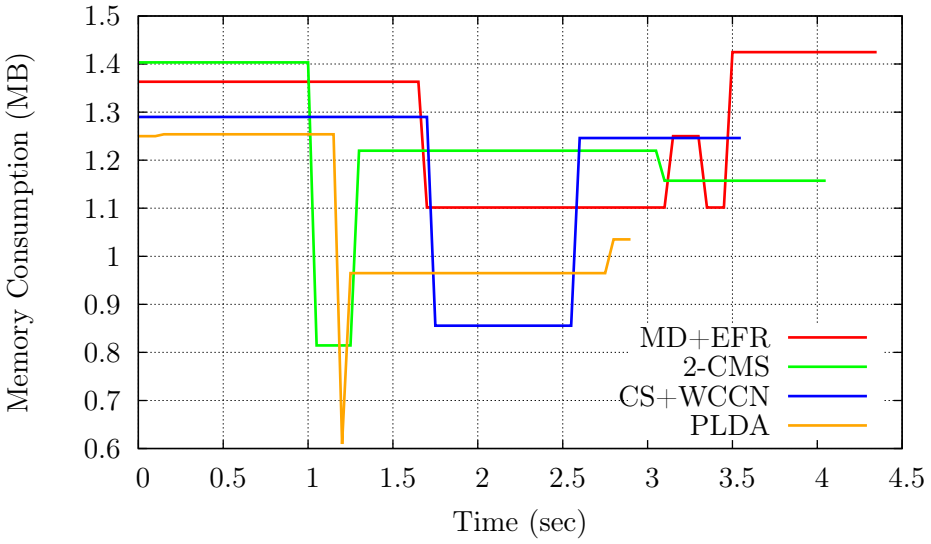
$$score = \frac{\int \mathcal{N}(w_1|y, W)\mathcal{N}(w_2|y, W)\mathcal{N}(y|\mu, B)\, dy}{\prod_{i-1,2} \int \mathcal{N}(w_i|y, W)\mathcal{N}(y|\mu, B)\, dy} \tag{5.10}$$

The complete solution of equation 5.10 can be found in [6].

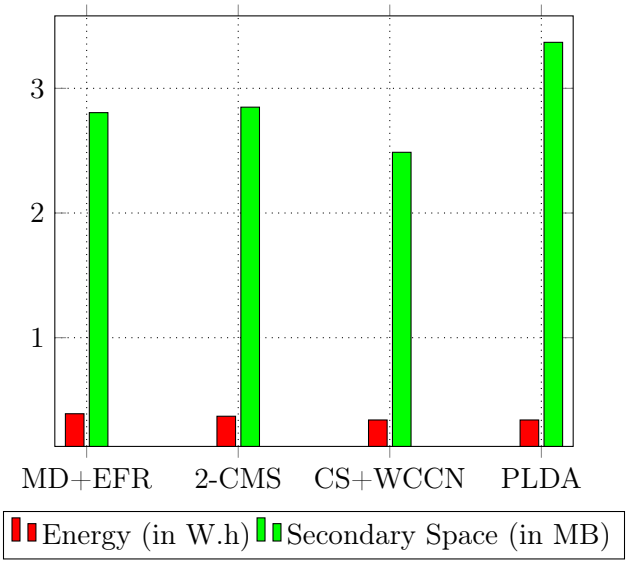
Figure 5.3a shows the memory consumption of i-vector based methods. Though the power consumption gain from PLDA method is not much as can be seen from 5.3, the speed of execution and memory requirements clearly show that PLDA is better than the other three methods in terms of resource usage.

TABLE 5.3: Resource consumption for different scoring methods when used with i-vectors for speaker recognition

Approach	**Memory Consumption (MB)**		**Energy**	**Space**	**Time**
	Average	**Maximum**	**(W.h)**	**(KB)**	**(sec)**
MD + EFR	1306.2	1459	0.39	2805	4.40
2-Covariance	1256.2	1437	0.37	2849	4.10
CS + WCCN	1203.4	1321	0.34	2487	3.60
PLDA	1106	1284	0.34	3369	2.95



(A) Memory Consumption



(B) Energy and Space Requirements

FIGURE 5.3: Resource consumption for different scoring methods when used with i-vectors for speaker recognition

TABLE 5.4: Resource consumption for GMM-UBM, JFA and i-vector approaches for speaker recognition

Approach	**Memory Consumption (MB)**		**Energy**	**Space**	**Time**
	Average	**Maximum**	**(W.h)**	**(KB)**	**(sec)**
GMM-UBM	1291.5	1310	0.36	1013	8.65
JFA	1292	1488	0.36	1034	6.45
i-vectors	1106	1284	0.34	3369	2.95

5.7 Comparison of Methods

The graph in Figure 5.4a shows the variation of memory usage with time. It can be clearly seen that i-vector approach is much faster as compared to the JFA and GMM-UBM approach. Also the memory requirements are relatively lesser than the other two. The maximum and average memory used during the experiment is also reported in Table 5.5.

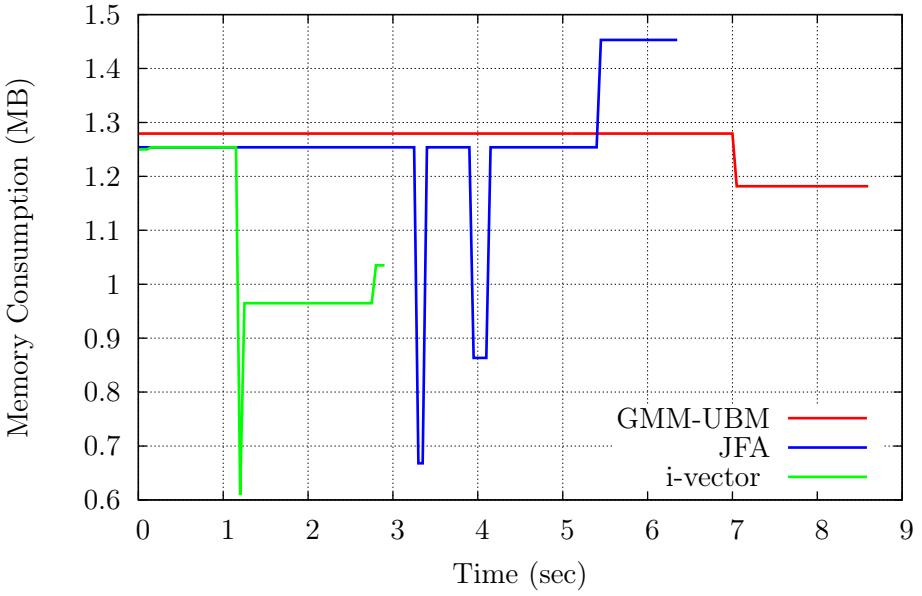
It should be noted that EML gives better results on parallelizable programs. JFA and GMM-UBM approaches implemented in the experiments do not have any preprocessing to be done, whereas in the i-vector approach normalization was done prior to actual scoring. Hence the results for i-vector approach might be improved if power consumption calculation is done without any parallelization.

The space requirements of the approaches keep increasing as we move from GMM-UBM to i-vector owing to the large code size and complexity for i-vector calculation leading to larger executables.

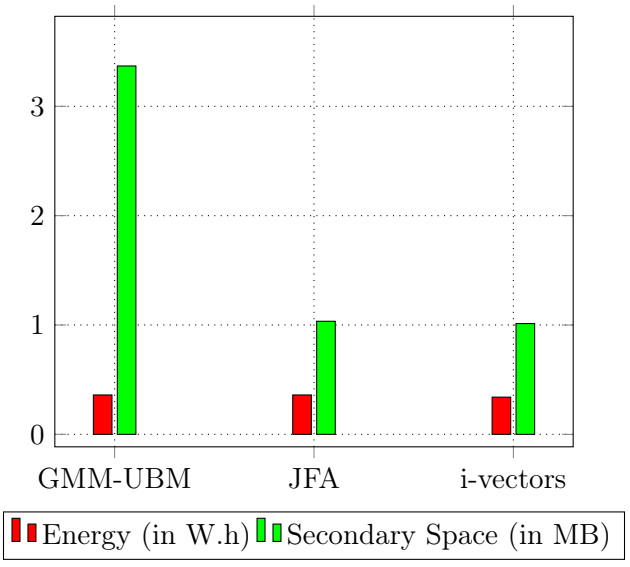
5.8 Accuracy Comparison

Other than resource consumption, accuracy of the different methods proposed in this chapter are to be considered before drawing any final conclusion. The accuracy of the methods based on various datasets is summarized in Table 5.5. The accuracies are calculated as the percentage of speakers the method is able to identify accurately. The inaccuracies are reported when the confidence values generated after the scoring step are within 2% of each other for multiple speakers. Hence in such a case the method cannot tell the test utterance is associated with which recording with sufficient confidence.

From the data it can be seen that i-vector-PLDA method performs better than others in terms of accuracy as expected. Only i-vector-CMS methods gives comparable accuracy, but owing to its large resource usage i-vector-PLDA method must be preferred.



(A) Memory Consumption



(B) Energy and Space Requirements

FIGURE 5.4: Resource consumption for GMM-UBM, JFA and i-vector approaches for speaker recognition

TABLE 5.5: Accuracy for GMM-UBM, JFA and i-vector approaches for speaker recognition with different scoring techniques using The MIT MDSVC Corpus

Recognition Technique	**Accuracy (%)**
GMM + T norm	87.50
GMM + Z norm	87.50
GMM + ZT norm	89.58
JFA + Dot scoring	93.75
i-vector + MD + EFR	91.67
i-vector + 2-Covariance	97.91
i-vector + CS + WCCN	95.83
i-vector + PLDA	97.91

5.9 Discussion

On the basis of these experiments we observed that the i-vector approach performs significantly better in terms of speed. The power consumption of all 3 approaches is comparable with i-vector approach performing slightly better. The i-vector technique also exhibit slight optimality in terms of accuracy and memory consumption but due to its large code-base it uses the maximum space to the order of thrice the space used by the other 2 methods.

Chapter 6
Conclusion

In this work, a comparison of the current state of the art techniques in terms of their resource utilization targeted towards speaker recognition is conducted. Experiments were performed on a standard corpus collected using mobile devices in a noisy environment. This dataset emulates the data likely to be encountered in real life environments. The algorithms tested using this dataset reflects the true nature of performance in actual situations.

The results show that the i-vector approach is more efficient in terms of memory usage and power consumption, which are the major focus areas for today's computing environments. Specifically in case of mobile applications, these two factors can play an important role in the success and usability of such software. Even in terms of running time the i-vector approach outperforms the other two approaches.

The current work only analyzes the resource usage on laptops and desktops in the lab environment. This can be extended to mobile devices in future to get more specific readings for various mobile platforms.

Future enhancements to this work include incorporating these experiments on heterogeneous operating systems like Android™, Windows® Phone, etc. as well as experimenting with different sized feature vectors for all these speaker recognition approaches.

Appendix A
Baum Welch Statistics

Given an speaker s and feature vectors $Y_1, Y_2, \ldots$. For each mixture component c, the Baum-Welch Statistics is defined as:

$$N_c(s) = \sum_t \gamma_t(c) \tag{A.1}$$

$$F_c(s) = \sum_t \gamma_t(c)\ Y_t \tag{A.2}$$

$$S_c(s) = diag\big(\sum_t \gamma_t(c)\ Y_t\, Y_t^*\big) \tag{A.3}$$

where, for each time $t, \gamma_t(c)$ is the posterior probability of the event that the feature vector Y_t is accounted for by the mixture component c. These posteriors are calculated using the UBM.

The centralized first and second order Baum-Welch statistics are denoted by $\widetilde{F}_c(s)$ and $\widetilde{S}_c(s)$.

$$\widetilde{F}_c(s) = \sum_{\mathrm{t}} \gamma_t(c).(Y_t - m_c) \tag{A.4}$$

$$\widetilde{S}_c(s) = diag\big(\sum_{\mathrm{t}} \gamma_t(c).(Y_t - m_c).(Y_t - m_c)^*\big) \tag{A.5}$$

where, m_c is the subvector of m corresponding to the mixture component c.

Appendix B
ALIZE 3.0

The ALIZE Toolkit [22] provides a wide variety of methods for complete i-vector extraction and processing. It has a full set of methods for normalization and scoring. It is written in C++ and provides implementations for JFA. It is provided under LGPL licence. ALIZE consists of two major packages:

- ALIZE package: It is a low level library that consists of all the methods related to Gaussian mixtures.
- LIA_RAL package: It is a high level package aimed at providing programs related to Automatic Speaker Detection. It consists of the following parts:
  - LIA_SpkTools: It is a high level library containing finished tools provided by LIA_RAL.
  - LIA_SpkDet: It consists of tools required for any task related to biometric authentication system.
  - LIA_Utils: It contains tools required in changing ALIZE related formats like GMM models, any parameters, etc.
  - LIA_SpkSeg: It is a recent tool added to ALIZE toolkit which is to be used for Speaker Diarization.

ALIZE is one of the most complete set of libraries that are required to develop a speech recognition application. Other than GMM-UBM model, this toolkit also consists of implementation for Joint Factor Analysis and i-vectors. The wide range of scoring methods inbuilt in the system are also sufficient for most practical systems. Though ALIZE provides nearly all the methods required for performing any kind of speech recognition, it is difficult to start with due to insufficient documentation provided with the toolkit.

The contents of this appendix can be found on ALIZE wiki[1]. It is included here for the purpose of completeness.

B.1 Binaries

1. Feature Extraction: It is done using `sfbcep` utility. It can extract 19 MFCC + log-energy coefficient + first and second order delta coefficients.

[1] http://mistral.univ-avignon.fr/mediawiki/index.php/

2. File Conversion: `NormFeat` and `EnergyDetector` used at this step to perform the initial preprocessing. It include windowing, silence removal (Voice Activity Detection), feature normalization, etc.

3. Technique Specific Tweaking:

   (a) **GMM-UBM**: Use `ComputeTest` and `ComputeNorm` utilities to do scoring and compute the norms respectively. For changing the Norm type, options should be changed in configuration file as shown in Table B.1. `targetIdList`, `ndxFileName` and `outputFileName` will be changed in configuration file used with `ComputeTest` according to the `normType` chosen.
   Usage: `ComputeTest --config <ComputeTest_configFileName>`
   `ComputeNorm --config <ComputeNorm_configFileName>`

Table B.1: Options in ALIZE for changing Norm Type with `ComputeNorm`

Norm Type	**Value of** `normType`	**Other Variables to change**
Z-norm	`znorm`	`znormNistFile`, `outputFileBaseName`
T-norm	`tnorm`	`tnormNistFile`, `outputFileBaseName`
ZT-norm	`ztnorm`	`znormNistFile`, `tnormNistFile`, `ztnormNistFile`, `outputFileBaseName`

   (b) **JFA**: Stepwise flow for performing Joint Factor Analysis is given in Table B.2. The command format to be followed when calling binaries will be:
   `<binary_name> --config <configFileName>`

Table B.2: Steps for performing JFA using ALIZE and corresponding binaries

Step	**Binary Name**	**Function**
1	`TrainWorld`	Train UBM by EM algorithm
2	`EigenVoice`	Estimate the EigenVoice Matrix
3	`ComputeJFAStats`, `EigenChannel`	Compute sufficient statistics for eigenchannel training
4	`EstimateDMatrix`	Compute sufficient statistics for DMatrix Training
5	`TrainTarget`	Train Speaker Dependent GMMs
6	`ComputeTest`	Compute Likelihood (Scoring)

   (c) **i-vectors**: Stepwise flow for using i-vectors with ALIZE is given in Table B.3. The command format to be followed when calling binaries will be:
   `<binary_name> --config <configFileName>`

   Use of `IvTest` is explained in the section **B.2** owing to large set of options available to use with it.

B.2 Scoring Methods

There are multiple scoring methods available to be used with i-vectors in ALIZE toolkit. To choose the proper scoring technique value of the variable `scoring` needs to be changed

Table B.3: Steps for using i-vectors with ALIZE and corresponding binaries

Step	Binary Name	Function
1	`TrainWorld`	Train UBM by EM algorithm
2	`TotalVariability`	Train TotalVariability matrix
3	`IvExtractor`	Extract i-vectors
4	`IvTest`	Scoring

in the configuration file. The possible values are listed in Table B.4. Depending on the value of `scoring`, some more options needs to be configured in the same configuration file. The options that can be configured with each type of scoring are listed in different tables whose references are also provided in Table B.4.

Table B.4: Options in ALIZE for changing Scoring Technique with `IvTest`

Scoring Technique	Value of `scoring`	Other Variables to change
Cosine Scoring + WCCN	`cosine`	Refer Table B.5
Mahanalobis Scoring + EFR	`mahanalobis`	Refer Table B.6
2-Covariance	`2cov`	Refer Table B.7
PLDA	`plda`	Refer Table B.8

Table B.5: Options in ALIZE for Cosine Similarity Scoring

Option	Value	Usage
wccn	boolean	apply Within Class Covariance Normalization
ivNormIterationNb	integer	number of iteration for normalization
loadWccnMatrix	true / false	if false then compute the WCCN matrix, if true, load it from file
wccnMatrix	string	name of the WCCN matrix file to load or save. Default value is WCCN
backgroundNdxFilename	string	list of files and class definitions for WCCN or LDA matrix computation

Table B.6: Options in ALIZE for Mahanalobis Scoring

Option	Value	Usage
backgroundNdxFilename	filename	list of files and class definitions for Mahalanobis matrix computation
mahalanobisMatrix	filename	filename of the Mahalanobis matrix to load or save
loadMahalanobisMatrix	true / false	load the matrix from file, if false the Mahalanobis matrix is estimated first

TABLE B.7: Options in ALIZE for Two-Covariance model

Option	**Value**	**Usage**
load2covMatrix	true / false	load matrices from files, if false the meta-parameters of the two-covariance model are estimated first
backgroundNdxFilename	filename	list of files and class definitions for two-covariance matrices computation
TwoCovFilename	filename	root of the W and B matrices used to define the two co-variance model. Filenames are obtained by adding "_W" and "_B" to the "TwoCovFilename" string. Default values is "2Cov"

TABLE B.8: Options in ALIZE for PLDA

Option	**Value**	**Usage**
pldaLoadModel	true / false	load the PLDA model from files, if false the meta-parameters are estimated first
pldaLoadInitMatrices	true / false	in case the PLDA model is estimated, load matrices for initialization. If false, the matrices are initialized randomly
iVectSize	integer	dimension of the i-vector for matrix initialization
pldaEigenVoiceNumber	integer	rank of the EigenVoice matrix
pldaEigenChannelNumber	integer	rank of the EigenChannel matrix
backgroundNdxFilename	filename	list of files and class definitions for PLDA model computation
pldaNbIt	integer	number of iterations to perform for PLDA parameters estimation
pldaScoring	native / enrollMean	type of score to compute (see dedicated section for more information)
pldaEigenVoiceMatrix	filename	EigenVoice matrix to load or save
pldaEigenChannelMatrix	filename	EigenChannel matrix to load or save
pldaSigmaMatrix	filename	Noise covariance matrix to load or save
pldaMeanVec	filename	mean vector of the development set
pldaMinDivMean	filename	mean vector re-estimated during PLDA model training
pldaEigenVoiceMatrixInit	filename	EigenVoice matrix to load for initialization
pldaEigenChannelMatrixInit	filename	eigenChannel matrix to load for initialization
pldaSigmaMatrixInit	filename	noise covariance matrix to load for initialization
pldaMeanVecInit	filename	mean vector to load for initialization

Bibliography

[1] Aronowitz, H., Barkan, O., 2012. Efficient approximated i-vector extraction, in: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, pp. 4789–4792. doi:10.1109/ICASSP.2012.6288990.

[2] Bourdon, A., Noureddine, A., Rouvoy, R., Seinturier, L., 2013. PowerAPI: A Software Library to Monitor the Energy Consumed at the Process-Level. ERCIM News 92, 43–44.

[3] Bousquet, P., Matrouf, D., Bonastre, J., 2011. Intersession Compensation and Scoring Methods in the i-vectors Space for Speaker Recognition, in: INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011, pp. 485–488. URL: http://www.isca-speech.org/archive/interspeech_2011/i11_0485.html.

[4] Bridle, J., Brown, M., 1974. An Experimental Automatic Word Recognition System. JSRU Report 1003.

[5] Browne, S., Dongarra, J., Garner, N., Ho, G., Mucci, P., 2000. A Portable Programming Interface for Performance Evaluation on Modern Processors. Int. J. High Perform. Comput. Appl. 14, 189–204. URL: http://dx.doi.org/10.1177/109434200001400303, doi:10.1177/109434200001400303.

[6] Brümmer, N., de Villiers, E., 2010. The Speaker Partitioning Problem, in: Odyssey 2010: The Speaker and Language Recognition Workshop, Brno, Czech Republic, June 28 - July 1, 2010, p. 34. URL: http://www.isca-speech.org/archive_open/odyssey_2010/od10_034.html.

[7] Burget, L., Brummer, N., Reynolds, D., Kenny, P., Pelecanos, J., Vogt, R., Castaldo, F., Dehak, N., Dehak, R., Glembek, O., Karam, Z., Jr., J.N., Na, E., Costin, C., Hubeika, V., Kajarekar, S., Scheffer, N., Cernocky, J., 2008. Robust speaker recognition over varying channels. Technical Report.

[8] Cabrera, A., Almeida, F., Arteaga, J., Blanco, V., 2014. Measuring energy consumption using EML (Energy Measurement Library). Computer Science - Research and Development , 1–9doi:10.1007/s00450-014-0269-5.

[9] Dehak, N., 2009. Discriminative and Generative Approaches for Long- and Short-term Speaker Characteristics Modeling: Application to Speaker Verification. Ph.D. thesis. Ecole de Technologie Superieure (Canada). AAINR50490.

[10] Dehak, N., Karam, Z.N., Reynolds, D.A., Dehak, R., Campbell, W.M., Glass, J.R., 2011a. A channel-blind system for speaker verification, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,

ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic, pp. 4536–4539. doi:10.1109/ICASSP.2011.5947363.

[11] Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., Ouellet, P., 2011b. Front-End Factor Analysis for Speaker Verification. Audio, Speech, and Language Processing, IEEE Transactions on 19, 788–798. doi:10.1109/TASL.2010.2064307.

[12] Dehak, N., Shum, S., 2011. Low-dimensional speech representation based on Factor Analysis and its applications. Johns Hopkins CLSP Lecture .

[13] Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the royal statistical society. Series B (methodological) , 1–38.

[14] Glembek, O., Burget, L., Brümmer, N., Plchot, O., Matejka, P., 2011. Discriminatively Trained i-vector Extractor for Speaker Verification, in: INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011, pp. 137–140. URL: http://www.isca-speech.org/archive/interspeech_2011/i11_0137.html.

[15] Glembek, O., Burget, L., Dehak, N., Brummer, N., Kenny, P., 2009. Comparison of scoring methods used in speaker recognition with Joint Factor Analysis, in: Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on, pp. 4057–4060. doi:10.1109/ICASSP.2009.4960519.

[16] Gupta, M., Verma, P., Bhattacharya, T., Das, P.K., 2015. A mobile agents based distributed speech recognition engine for controlling multiple robots, in: Proceedings of the 2015 Conference on Advances In Robotics (Accepted).

[17] Jiang, Y., Lee, K., Tang, Z., Ma, B., Larcher, A., Li, H., 2012. PLDA Modeling in I-Vector and Supervector Space for Speaker Verification , in: INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012, pp. 1680–1683. URL: http://www.isca-speech.org/archive/interspeech_2012/i12_1680.html.

[18] Kenny, P., 2005. Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms. Technical Report.

[19] Kenny, P., Boulianne, G., Ouellet, P., Dumouchel, P., 2007a. Joint Factor Analysis Versus Eigenchannels in Speaker Recognition. Audio, Speech, and Language Processing, IEEE Transactions on 15, 1435–1447. doi:10.1109/TASL.2006.881693.

[20] Kenny, P., Boulianne, G., Ouellet, P., Dumouchel, P., 2007b. Speaker and Session Variability in GMM-Based Speaker Verification. Audio, Speech, and Language Processing, IEEE Transactions on 15, 1448–1460. doi:10.1109/TASL.2007.894527.

[21] Kenny, P., Ouellet, P., Dehak, N., Gupta, V., Dumouchel, P., 2008. A Study of Interspeaker Variability in Speaker Verification. Audio, Speech, and Language Processing, IEEE Transactions on 16, 980–988. doi:10.1109/TASL.2008.925147.

[22] Larcher, A., Bonastre, J., Fauve, B.G.B., Lee, K., Lévy, C., Li, H., Mason, J.S.D., Parfait, J., 2013. ALIZE 3.0 - Open Source Toolkit for State-of-the-Art Speaker Recognition , in: INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013, pp. 2768–2772. URL: http://www.isca-speech.org/archive/interspeech_2013/i13_2768.html.

[23] McLachlan, G., 1988. Mixture Models. Marcel Dekker, New York, NY.

[24] Noureddine, A., Rouvoy, R., Seinturier, L., 2013. A Review of Energy Measurement Approaches. SIGOPS Oper. Syst. Rev. 47, 42–49. URL: http://doi.acm.org/10.1145/2553070.2553077, doi:10.1145/2553070.2553077.

[25] Noureddine, A., Rouvoy, R., Seinturier, L., 2014. Unit Testing of Energy Consumption of Software Libraries, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, New York, NY, USA. pp. 1200–1205. URL: http://doi.acm.org/10.1145/2554850.2554932, doi:10.1145/2554850.2554932.

[26] Reynolds, D., 2009a. Gaussian Mixture Models, in: Li, S., Jain, A. (Eds.), Encyclopedia of Biometrics. Springer US, pp. 659–663. URL: http://dx.doi.org/10.1007/978-0-387-73003-5_196, doi:10.1007/978-0-387-73003-5_196.

[27] Reynolds, D., 2009b. Universal Background Models, in: Li, S., Jain, A. (Eds.), Encyclopedia of Biometrics. Springer US, pp. 1349–1352. URL: http://dx.doi.org/10.1007/978-0-387-73003-5_197, doi:10.1007/978-0-387-73003-5_197.

[28] Reynolds, D.A., 1995. Speaker Identification and Verification using Gaussian Mixture Speaker Models. Speech Communication 17, 91–108. doi:10.1016/0167-6393(95)00009-D.

[29] Reynolds, D.A., Quatieri, T.F., Dunn, R.B., 2000. Speaker Verification using Adapted Gaussian Mixture Models. Digital signal processing 10, 19–41.

[30] Shankar Mysore, A., Yaligar, V.S., Arrieta Ibarra, I., Simoiu, C., Goel, S., Arvind, R., Sumanth, C., Srikantan, A., HS, B., Pahadia, M., Dobha, T., Ahmed, A., Shankar, M., Agarwal, H., Agarwal, R., Anirudh-Kondaveeti, S., Arun-Gokhale, S., Attri, A., Chandra, A., Chilukur, Y., Dharmaji, S., Garg, D., Gupta, N., Gupta, P., Jacob, G.M., Jain, S., Joshi, S., Khajuria, T., Khillan, S., Konam, S., Kumar-Kolla, P., Loomba, S., Madan, R., Maharaja, A., Mathur, V., Munshi, B., Nawazish, M., Neehar-Kurukunda, V., Nirmal-Gavarraju, V., Parashar, S., Parikh, H., Paritala, A., Patil, A., Phatak, R., Pradhan, M., Ravichander, A., Sangeeth, K., Sankaranarayanan, S., Sehgal, V., Sheshan, A., Shibiraj, S., Singh, A., Singh, A., Sinha, P., Soni, P., Thomas, B., Varma-Dattada, K., Venkataraman, S., Verma, P., Yelurwar, I., 2015. Investigating the "Wisdom of Crowds" at scale, in: Under submission.

[31] Tiwari, V., Malik, S., Wolfe, A., 1994. Power Analysis of Embedded Software: A First Step Towards Software Power Minimization. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 2, 437–445. doi:10.1109/92.335012.

[32] Verma, P., Das, P.K., 2015a. A comparative study of resource usage for speaker recognition techniques, in: Under submission.

[33] Verma, P., Das, P.K., 2015b. i-Vectors in speech processing applications: A Survey. International Journal of Speech Technology (Accepted) .

[34] Verma, P., Gupta, M., Bhattacharya, T., Das, P.K., 2014. Improving services using mobile agents-based IoT in a smart city, in: 2014 International Conference on Contemporary Computing and Informatics, IEEE. p. 107–111. doi:10.1109/IC3I.2014.7019766.

[35] Wolf, J.J., 1972. Efficient acoustic parameters for speaker recognition. The Journal of the Acoustical Society of America 51.

[36] Woo, R., Park, A., Hazen, T., 2006. The MIT Mobile Device Speaker Verification Corpus: Data Collection and Preliminary Experiments, in: Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006: The, pp. 1–6. doi:10.1109/ODYSSEY.2006.248083.

[37] Zheng, R., Zhang, S., Xu, B., 2005. A Comparative Study of Feature and Score Normalization for Speaker Verification, in: Zhang, D., Jain, A. (Eds.), Advances in Biometrics. Springer Berlin Heidelberg. volume 3832 of *Lecture Notes in Computer Science*, pp. 531–538. doi:10.1007/11608288_71.

List of Publications

National and International Conferences

- **Pulkit Verma**, Mayank Gupta, Tuhin Bhattacharya and Pradip K. Das, **Improving Services using Mobile Agents-based IoT in a Smart City**, 2014 International Conference on Contemporary Computing and Informatics, pp. 107-111. November 27-29, 2014, Mysore, India. [34]
  Paper URL: http://dx.doi.org/10.1109/IC3I.2014.7019766

- **Pulkit Verma** and Pradip K. Das, **i-Vectors in Speech Processing Applications: A Survey.**, International Journal of Speech Technology, 2015. Springer U.S. (Accepted) [33]
  Journal URL: https://www.springer.com/journal/10772

- Mayank Gupta, **Pulkit Verma**, Tuhin Bhattacharya and Pradip K. Das, **A Mobile Agents based Distributed Speech Recognition Engine for Controlling Multiple Robots**, 2nd International Conference on Advances In Robotics, 2015. (Accepted) [16]
  Conference URL: http://www.advancesinrobotics.com/

Papers Submitted

- **Pulkit Verma** and Pradip K. Das, **A Comparative Study of Resource Usage for Speaker Recognition Techniques**, 18th International Conference on Text, Speech and Dialogue 2015. Springer International Publishing. [32]
  Conference URL: http://www.kiv.zcu.cz/tsd2015/

- Alok Shankar Mysore et al., **Investigating the "Wisdom of Crowds" at Scale**, 28th Annual ACM Symposium on User Interface Software & Technology, 2015. [30]
  Conference URL: https://uist.acm.org/uist2015/about