

A REST API approach to Distributed Speaker Verification using Spoken Numeric Digits

Kevin Brennan¹ and Denis Flynn²

¹ Waterford Institute of Technology, Cork Road, Waterford, Ireland
(email:kb@ping101.com)

² Waterford Institute of Technology, Cork Road, Waterford, Ireland
(email:mdflynn@wit.ie)

Abstract. Typically, speech verification uses a centralised architecture for all steps in speech processing, namely; feature extraction, feature selection, pattern matching and decision making. A distributed speaker verification system decouples the architecture to reduce channel effects, improve resource utilisation and increase privacy for remote parties. This paper considers an investigation into the effectiveness of a REST API approach to distributed speaker verification using spoken numeric digits (as is typical of PIN code authentication, or a card number for payment transaction). More specifically the decoupling of *feature extraction and selection* from *pattern matching and decision making* in the verification system. The paper suggests a practical implementation with sampling to demonstrate findings.

Keywords: ASR, ASV, REST API, Web API, speaker recognition, speaker verification, speaker authentication, text-dependant, feature extraction, feature selection, pattern matching, speaker models, cepstrum, mel-frequency, MFCC (Mel-Frequency Cepstral Coefficients), GMM (Gaussian Mixture Model), spectrogram, unwitting speakers, short utterance, Fourier transform, FFT, telephone, digits.

1 Introduction

1.1 Background.

Applications designed to recognise a human's voice, known as ASR (Automatic Speaker Recognition) systems, are typically used for speaker verification (authentication) and/or identification (the nuanced difference is discussed later in section 2.2). Generally, ASR systems centralise 'speech processing' architecture, even when serving remote internet clients. This centralised approach to processing architecture may be advantageous for some demanding ASR applications but for remote clients this close-

coupling leads to reduced privacy and sub-optimal resource usage. As an alternative, this paper considers how many ASR applications, especially those targeted at remote users, may successfully split the ‘speech processing’ process between client and server. The paper demonstrates the concept of de-centralised speaker verification using a short sequence of spoken digits implemented using a REST API (A common lightweight method of exposing Web services).

1.2 Problem Statement and Motivation

Recent advances in computing power, AI (Artificial Intelligence) and cloud computing have brought speech synthesis, speech recognition and speaker recognition to common use (SIRI, Amazon Alexa etc.). Many next-generation ‘machines’ will feature human speech interaction, and for some the prime user-interface will be voice.

Apart from the user-interface challenge voice-machine interaction brings, we must also consider how this technology shift will impact resources and transaction security.

Consider, for example, a voice-to-machine e-commerce transaction. Since the complete transaction is spoken, including payment authorisation, the user authentication process must differ from a ‘text approach’. Both consumer and merchant prefer a ‘frictionless’ transaction, thus requiring authentication steps to be kept to a minimum, or preferably hidden completely. For this to happen, background authentication must occur, i.e. without additional prompting. One possible way to achieve this is using the transient ‘transaction speech’ to verify the customers identity.

Most current methods of speaker recognition use a centralised architecture, meaning raw speech data is presented directly to the remote ASR (Automatic Speaker Recognition) system for verification or identification to occur. The Web API’s used for speaker recognition today post binary speech data (Microsoft n.d.) in their payload. This approach poses two problems. Firstly, resource utilisation is inefficient, for both bandwidth and remote CPU. And, secondly privacy rights issues arise when remotely storing, transmitting or processing the raw speech data. Part-processing speech at the client-side could mitigate both factors.

1.3 Research Objectives

This paper considers an investigation into the effectiveness of using a REST API approach to distributed speaker verification with a short series of spoken numeric digits (as is typical of PIN code authentication, or a card number for payment transaction).

The main objective of the research paper is to propose and demonstrate a REST API (A common lightweight method of exposing Web services) which successfully decouples *feature extraction and selection* from *pattern matching and decision making* in an ASR (Automatic Speaker Recognition) system. To keep within the timeframe the scope is limited to *Speaker Verification* of a short series of spoken numeric digits (*text-depend*

short utterance). However, the work equally applies in the broader context of Speaker & Speech Recognition; it's hoped the work stimulates further development of the findings.

1.4 Research Questions

1. Can a distributed speaker verification REST API, which decouples *feature extraction and selection* from *feature vector matching and decision making*, be used to verify speakers using a short sequence of spoken numeric digits?
2. Can a regular web browser client successfully perform real-time speech *feature extraction and selection* for a distributed speaker verification REST API?
3. What is the real-world verification performance of the distributed speaker verification REST API?

1.5 Scope & Contribution.

Research into Web API's for text dependent distributed speaker verification using numeric digits applies to a relatively narrow field within speech processing. However, the paper should lay the groundwork for further studies including multi-modal authentication, speaker identification and speaker recognition.

1.6 Conclusion

'Speech first' is the design approach taken for many new applications. However, existing speech API's take the rather primitive approach of sending unprocessed binary speech data. This has implications for resource usage and privacy. This paper investigates an alternative 'lighter' way to implement speech API's using text-dependent speaker verification to explore the boundaries of what's achievable.

2 Literature Review:

2.1 Introduction

Speech Processing is a diverse area within Signal Processing which includes: *Speaker Recognition* (who is saying it), *Speech Recognition* (what is being said) and *Language Identification* (what language is spoken). This paper is focussed on the specific type of *Speaker Recognition* known as *Speaker Verification* (often called *Speaker Authentication*), however the work equally applies to the broader field of *Speaker Recognition*.

Speaker Verification is the process of ***authenticating a speaker*** using their spoken voice. Essentially a speaker's voice is confirmed as matching a pre-recorded 'model' (this contrasts with *Speaker Identification* which 'finds' the correct speaker in a known set).

All types of *Speaker Recognition* utilise very similar processing techniques, that is **extracting ‘features’** from the voice followed by **feature analysis & decision making**. The differences between the main types of *identification* and *validation* are subtle, the literature review will therefore mostly refer to research in the more generalised context of *Speaker Recognition*.

For the sake of disambiguation *Speech Recognition* is the process of **word extraction** (*not* the subject of the paper), though similarities in *feature extraction* techniques may also exist. Therefore, sometimes the paper may also refer to research in this area where context is appropriate.

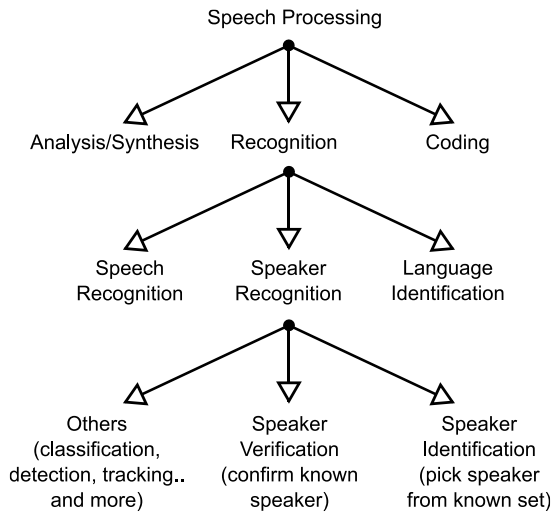


Fig. 1. Branches of Speech Processing. Adapted from (Campell 1997)

The study of *Speaker Recognition* has been active since the mid-20th century (Pollack, Pickett and Sumbly 1954). Early studies focussed on extraction of idiosyncratic features of speech with human listeners providing analysis. The need for an automatic approach to analysis was soon established and studies with pattern matching and statistical methods began a decade later (Pruzansky, Pattern-Matching Procedure for Automatic Talker Recognition 1963) (Pruzansky and Mathews, Talker-Recognition Procedure Based on Analysis of Variance 1963).

Early studies focussed on text-dependent analysis (matching a known phrase), which is less challenging than text-independent (matching an unknown phrase). By the 1970's automatic speakers systems appear which demonstrate 98% accuracy in verification of text-dependent speech from a laboratory recording (Atal 1974). Much of the subsequent speaker recognition studies focus on the challenges of text-independent recognition and optimising the speech model used for real world channel effects (variations in recordings).

However, two relevant studies deal with utterances composed of digits and their successful use for speaker recognition. A 1985 study using 10 random isolated digits (recorded over the telephone) resulted in over 98% speaker identification (Soong, et al. 1985). A 1991 study demonstrated that even with a short sequence of only four isolated digits (recorded over the telephone), a speaker could be verified with an average equal-error rate of less than 3 %. (Tisby 1991).

Driven by the limited bandwidth available early in mobile communications an ETSI standard for Distributed Speech Recognition systems was published in 2000 (ETSI 2000). Although the standard was designed for speech recognition it was successfully demonstrated to work in speaker verification in 2002. However, this standard was designed with embedded systems in mind, therefore purposely limiting client-side capabilities (Broun, et al. 2002).

In 2016, Microsoft launched a speaker recognition web API, this is not a distributed system verification system as speech data is posted as a binary file (Microsoft n.d.).

2.2 Speaker Recognition.

A human voice conveys different information. In addition to what we say, it conveys unspoken emotion and identity. *Speaker recognition* is concerned with extracting the identity information to decide who is speaking. To do this we use the idiosyncratic characteristics of speech, commonly known as *feature parameters* in the field of speaker recognition.

The process of speaker recognition takes extracted *feature parameter* and applies techniques for *matching and decision making*. The typical objective being *identification* or *verification* of an individual (others being detection or classification). *Verification* and *identification* employ a subtle difference in *matching and decision-making* techniques but are otherwise the same.

This paper is specifically concerned with *verification*. To explain the nuanced difference, *identification* is ‘picking’ the correct speaker from a group, whereas *verification* is confirming the authenticity of a specific speaker.

For example, consider you have a family member Alice. If Alice phones and says, “*who am I?*”, this is a process of *identification*. If Alice calls and says, “*I am Alice, am I Alice?*”, this is a process of *verification* (or authentication).

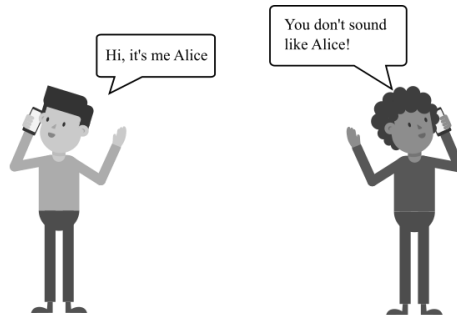


Fig. 2. Speaker Verification

Applications of speaker verification include: access control, telephone banking, and telephone card payments.

Speech Production.

To understand how speaker recognition works we begin with a basic understanding of speech production. This section reviews our understanding of speech production.

It may not be immediately obvious that characteristics we seek in speaker recognition (who said it) differ from speech recognition (what is said). Speaker recognition is concerned with identifying features that relate to the speaker's physical biology or some learned speech behaviour. (Campell 1997).

Speech production is best described in a two-step process. The first step consists of air from the lungs making a 'sound' as it's being forced across the vocal cords in the larynx. The second step is the 'sound' being changed as it passes through the cavities above the vocal cords, also known as the vocal tract (includes: nasal cavity, mouth, teeth, tongue, lips, pharynx).

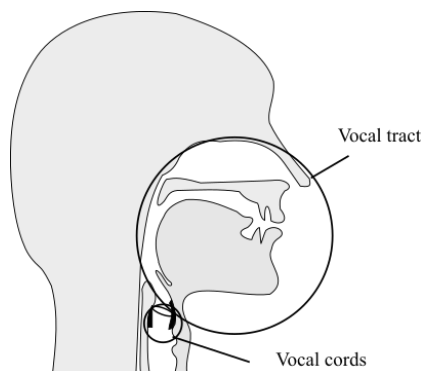


Fig. 3. Physiology of speech

This two-step process is known as the source-filter model (Fant 1960). The vocal cord ‘sound’ being the source, and the vocal tract acting as the filter.

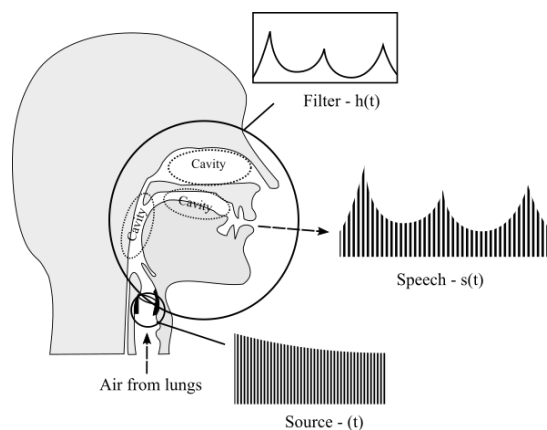


Fig. 4. The source-filter speech production model. Adapted from (SAWUSCH 2005)

The ‘pitch’ of the ‘source sound’ relates to the size and muscular tension of an individual’s vocal cords, it is therefore a physical characteristic useful in *speaker identification*. The vocal cords vibrate at the *fundamental frequency* (F_0) and a series of integer multiples (harmonics) (SAWUSCH 2005).

Further up, the vocal tract consists of nasal and oral cavities which resonate when the ‘source’ passes through it (much the same way a bottle makes a sound when you blow across its opening). This resonance is a significant contribution to the ‘filter’ and because the frequency produced relates to the size of the cavities, they are another useful physical characteristic for *speaker identification* (picture the change in pitch when you blow across bottles of different sizes). These vocal tract resonances are known as *formants* and can be identified in the frequency spectrum (**Fig. 4** above). (Campbell 1997).

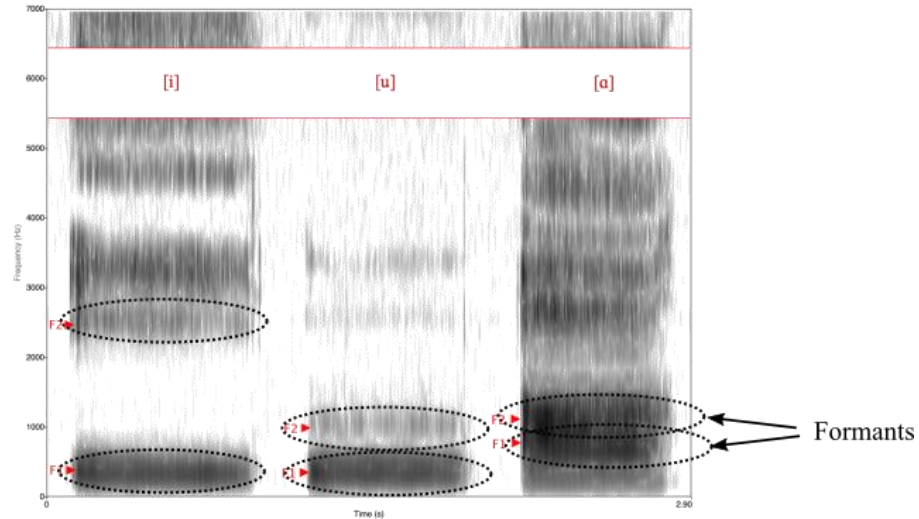


Fig. 5. Formants viewed on a spectrogram

Automatic Speaker Recognition.

ASR (Automatic Speaker Recognition) systems are applications developed to recognise a voice independent of human interaction. The technical challenges of unassisted recognition may be overlooked because humans perform this task so effortlessly (Hollien, Majewski and Hollien 1974). ASR (Automatic Speaker Recognition) systems that specialise in verification are sometimes referred to as ASV's (Automatic Speaker Verification) systems.

Side note: In the context of literature it's worth noting for disambiguation that ASR may also refer to Automatic Speech Recognition, which is not the same thing (Refer to **Fig. 1**. Branches of Speech Processing. Adapted from **Fig. 1** p.4)

However, automatic speaker recognition methods have many challenges to face, the most significant being the variation found in similar speech 'utterances' from the same person. These variations are mostly due to technology factors (background noise, poor microphone, sampling rate etc.) and natural speech variability (sickness, stress, emotion, dialect, reading etc.) (Hansen and Hasan 2015).

The general approach to speech recognition systems is illustrated in the figure below. The speech features of interest are extracted across time intervals, called feature vectors. The vectors are then matched against a known speaker model, then a decision is taken on similarity. There is normally a co-dependency between features, models and matching. Another way of putting this is that the matching of the feature extraction method, the model and the matching is important.

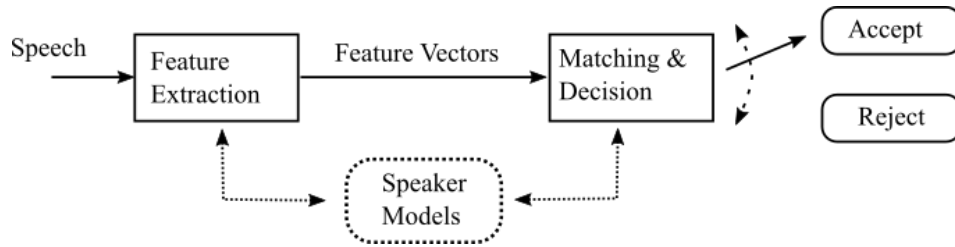


Fig. 6. General speaker recognition system. Adapted from (Campell 1997) (Hansen and Hasan 2015).

Feature Extraction.

For accurate speaker recognition we must be able to identify the idiosyncratic characteristics of speech (various characteristics highlighted in the last section). These are generally referred to as *feature parameters*. As this paper is interested in *short utterances* we are only concerned with *short-term features*, that's features available from a short interval recording. This section reviews mathematical methods used to process speech and identify *short-term features*.

Speech acquisition.

Before speech can be processed, it must be in an accessible form. This acquisition process itself can introduce difficulties which will affect feature identification.

Sound is a pressure wave which must first be converted to the digital domain. This process involves a microphone, conditioning, and then analogue to digital conversion. The analogue to digital conversion process requires samples to be taken at a regular period, the rate of sampling directly affects the quality of the digital version. Lower sampling has the effect of filtering higher frequencies and introducing aliasing (if not correctly pre-filtered). Since most of the linguistic information we require is in the band up to 5Khz, and the Nyquist rate determines we must sample at twice that rate, a sampling range of over 10Khz will suffice for accurate speaker recognition (SAWUSCH

2005). A loss of accuracy must be considered when processing speech recorded with lower bandwidth, such as a telephone (upper limit 3.4Khz).

Additionally, *channel effects* must be considered; these are the various acoustic mismatches which occur between similar recordings due to the characteristics of the channel such as echo, background noise, frequency limitation, loudness.

Fourier Transform.

Previously it was noted the *fundamental frequency* and *formants* are features of interest in speaker recognition, both occur in the frequency domain. Therefore, the ability to map sound samples from the time to frequency domain is clearly useful. The function to perform this mapping is the Fourier Transform (FT), proposed by Jean Baptiste Joseph Fourier (1768–1830). The Discrete Fourier Transform (DFT) is the equivalent function we must use for processing digital signals (SAWUSCH 2005). The Fourier Transform is used in Mel-frequency cepstral coefficient (MFCC) calculation (see below).

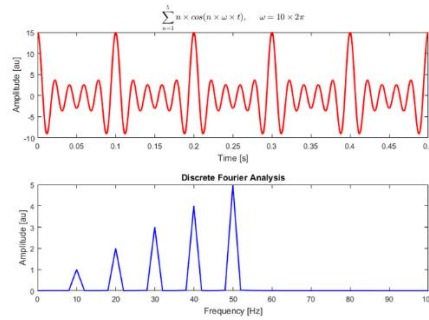


Fig. 7. Fourier Analysis (bottom) of a Cosine Summation Function (top).

Linear Prediction

Linear predictive coding (LPC) mathematically models the vocal tract to extract features. It uses a source-filter model, as described in section *speech production*. In LPC Speech production is treated as a source driving a set of resonators. Because LPC mathematically models the ‘filter’, it’s an effective method to extract formant features. Linear Predictive Cepstral Coefficients (LPCC) are a further derivation of the LPC spectral envelope which is also commonly used for feature extraction.

Cepstrum

Cepstral analysis is an alternative to LPC, it’s particularly effective at pitch extraction. Cepstral analysis of a signal results in a Cepstrum (a play on the word spectrum from which it’s derived). The Cepstrum is a result of the inverse Fourier analysis of the logarithm of the frequency spectrum (derived from Fourier transform).

Mel Frequency Cepstrum

The Mel Frequency Cepstrum (MFC) is an alternative feature extractor which considers the human auditory canal applied as a filter bank (using a perception scale of pitches known as the mel-scale). The calculated coefficients which make up the MFC, known as the mel-frequency cepstral coefficients (MFCCs) generally outperform LPC for feature extraction (Dhanalakshmi, Palanivel and Ramalingam 2011).

Other Features

Most classic features are based on LPC, LPCC, or MFCC. Other, but less common features extractors are; Log area ratio coefficients (LAR) , Line Spectral Frequencies (LSF), Discrete Wavelet Transform (DWT), Perceptual Linear Prediction (PLP), Cochlear Filter Cepstral Coefficients (CFCC), Linear and Exponential Frequency Cepstral Coefficients (LFCC and EFCC) and Gammatone Frequency Cepstral Coefficients (GFCC), Missing Feature Theory (MFT) & Local binary features (slice classifier) (Beigi, Speaker Recognition: Advancements and Challenges 2012).

Matching and Decision.

During a training phase extracted features must be characterised for an individual, this is known as modelling. Essentially the model must emphasis the feature set for an individual so comparison can take place. For the model to be robust it should also be capable of de-emphasising undesirable variations (see introduction to Automatic Speech Recognition above). Speaker verification is a result of observed feature vectors being compared with an individual's trained model (speaker identification would require comparison with all known models).

Matching the model and the feature vectors is essentially a pattern matching problem. Early approaches to this were *template based*, this is where observed feature vectors are considered a poor replica of the model and matching is finding the optimal alignment the two. Common template models are:

- Vector Quantization (VQ)
- Dynamic Time Warping (DTW)

Later approaches use *stochastic models* where matching is probabilistic. In this case, it's considered if the observed feature vectors are a likely result of the model (Campell 1997). Common stochastic models are;

- Hidden Markov Model (HMM)
- Gaussian Mixture Model (GMM)
- GMM Supervector Model with Support Vector Machines (GMM-SVM)
- Neural Network (many variants)
- Joint Factor Analysis (JFA)
- I-Vector

I-vectors are the current state-of-the-art in speaker verification modelling but it should be noted that neural network models have also had some degree of success in speaker identification, especially where channel conditions may be learned (Hansen and Hasan 2015).

2.3 Conclusion

Research into speaker recognition began in the 1950's and has continued at a steady pace. Successful automatic recognition techniques for text-dependent are well known. Modern speaker recognition research is generally focussed on improving models to counter channel variance. Distributed speaker recognition was explored in the early 2000's when phone and network resources were limited. Modern speech API's don't use a distributed model. This will be discussed in more detail in the following section.

3 Theory

3.1 Introduction

As reviewed in the literature, the theories and algorithms supporting automatic text-dependent speaker verification are well established but have never been applied to modern Web API's in a distributed fashion.

This paper puts forward the hypotheses that a REST API approach to distributed speaker verification using a short series of spoken numeric digits would be successful, and for remote clients such an API would improve security and make more efficient use of resources than existing centralised systems.

3.2 Distributed ASR Systems

The general model for an ASR system is outlined in the Literature Review section 2.2 **Speaker Recognition**. (reference **Fig. 6**, p.9). This consists of two main components: *feature extraction and selection* and *pattern matching and decision making*. The logical approach to remote de-coupling is a separation of these component parts, as illustrated in **Fig. 8** below. This is a verified approach taken in distributed ASR systems (ETSI 2000).

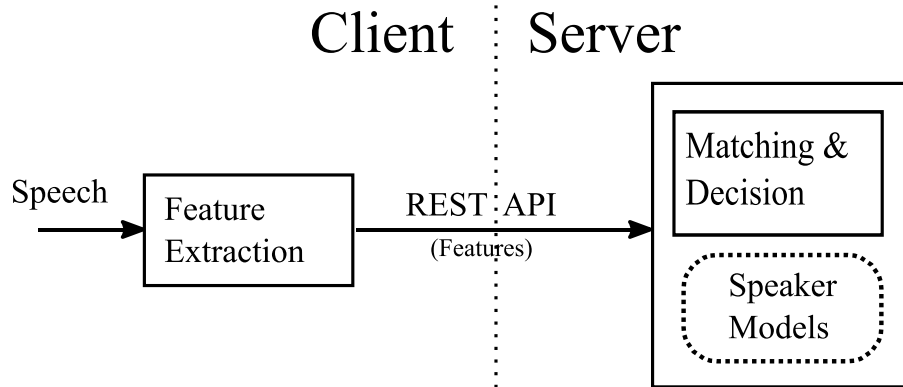


Fig. 8. De-coupling feature extraction from matching & decision making

3.3 Client Feature Extraction

The REST API client extracts the feature vectors necessary to describe the speakers voice characteristics.

The object of feature extraction is to remove all un-necessary information and only transmit that which is *needed* for the speaker's verification. Another way to look at this is we're attempting to remove any information a human does not use or perceive in speaker recognition. This process can also be viewed as a form of compression, or bandwidth reduction.

The extracted features are *Log Energy* and *Mel Frequency Cepstral Coefficient* (MFCC), both which have been used extensively in ASR systems.

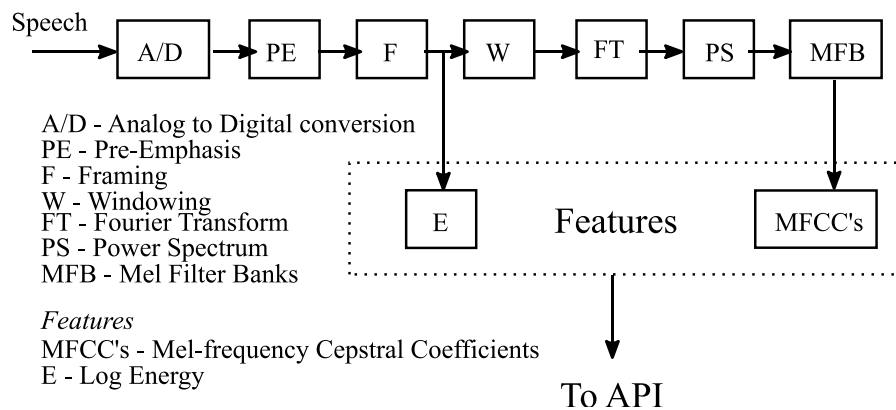


Fig. 9. Extraction of feature vectors

An explanation of the theory behind each step of feature extraction follows;

Analog to Digital conversion.

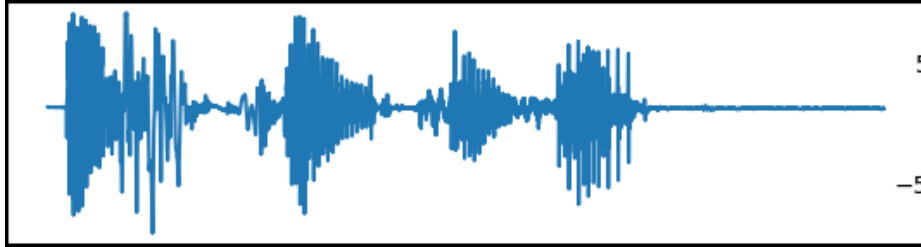


Fig. 10. Sampled speech signal

Pre-Emphasis

A pre-emphasis filter is applied to the source signal which amplifies the higher frequencies. This helps ‘balance’ the higher frequencies, which tend to have smaller magnitudes than lower frequencies, prior to the Fourier Transform.

$$y(t) = x(t) - \alpha x(t - 1)$$

Where α is the filter co-efficient (normally set to 0.97). Below **Fig. 11** shows **Fig. 10** (original signal) after pre-emphasis is applied.

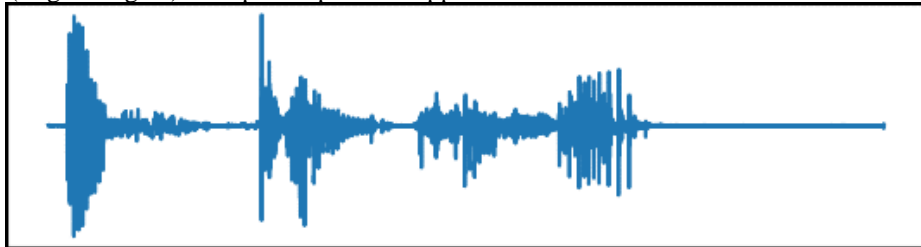


Fig. 11. Sampled signal after pre-emphasis

Framing

To transform the continuous speech signal into a useful frequency domain representation (using Fourier Transform) it first must be ‘chopped’ into shorter time frames. Framing for MFCC feature extraction is typically in 25ms intervals with a short overlap between frames.

The reasoning here is we’re looking for a time variant frequency domain representation of the whole signal. Conversely, if the FT was applied to the complete signal,

then the resultant frequency spectrum would be an aggregated view, which is of little use. **Fig. 12** below illustrates framing of the pre-emphasised signal.

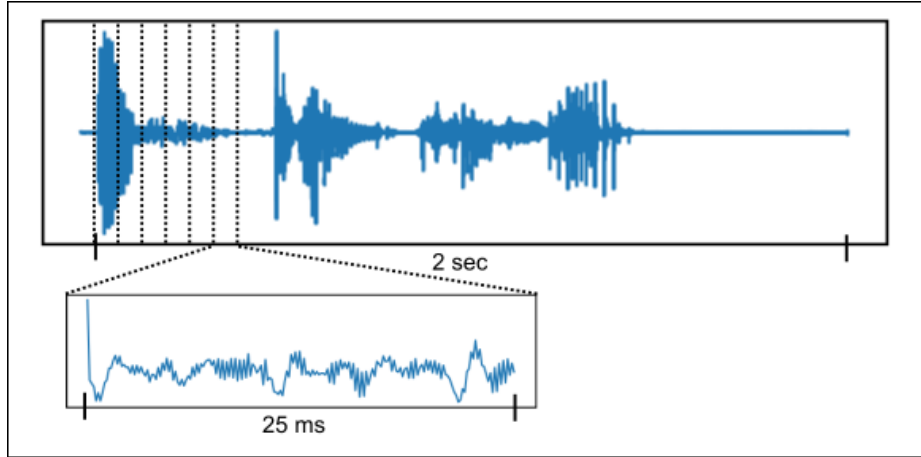


Fig. 12. Framing of the speech signal

Log Energy

The log energy is calculated for each frame and extracted for inclusion with the feature vectors. The log energy represents the ‘loudness’ of each frame. It’s therefore useful as a feature vector in speaker modelling and client voice activity detection.

$$\log E = \ln \left(\sum_{i=1}^N s(i)^2 \right)$$

Where, $0 \leq n \leq N - 1$, and N is the frame length and s is the input signal.

Windowing

A Hamming window is applied to each frame. The hamming window is a necessary pre-requisite to the Fourier Transform, which would otherwise represent the input as an infinite signal. Another way to put this is that without the Hamming window the frequency spectrum would include unwanted ripples which would not be truly part of the finite input. The Hamming window equation is;

$$w[n] = 0.54 - 0.46 \cos \left(\frac{2\pi n}{N-1} \right)$$

Where, $0 \leq n \leq N - 1$, and N is the frame length.

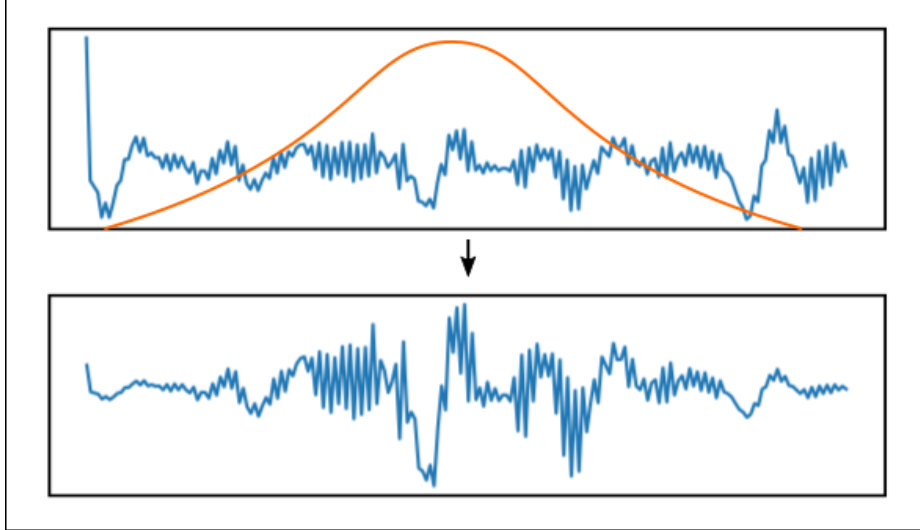


Fig. 13. Hamming window applied to a single frame.

Fourier Transform.

The Transform maps each frame into the frequency domain where most human speech perception occurs.

The Discrete Fourier Transform (DFT) algorithm is used as we are dealing with discrete samples. The equation for the DFT is shown below. Note that if a frames' size is not a power of two (256, 512..) the frame is zero padded before the DFT is applied to improve performance.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi kn}{N}}$$

Where, $0 \leq k \leq N - 1$, and N is the frame length.

The following **Fig. 14** illustrates the DFT applied to a single frame (per **Fig. 13**), and following that the DFT applied across the complete signal in **Fig. 15**

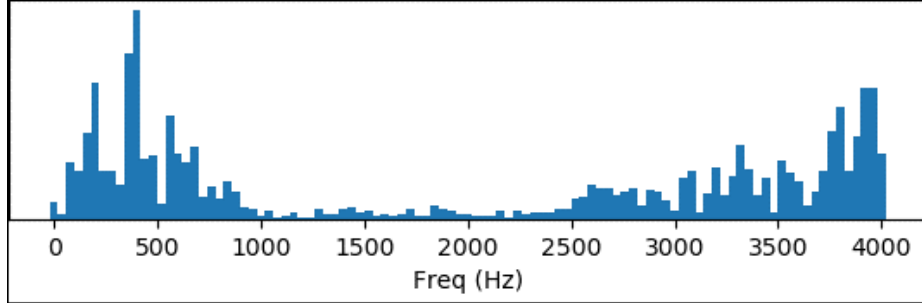


Fig. 14. Fourier Transform of a single frame (Frequency →)

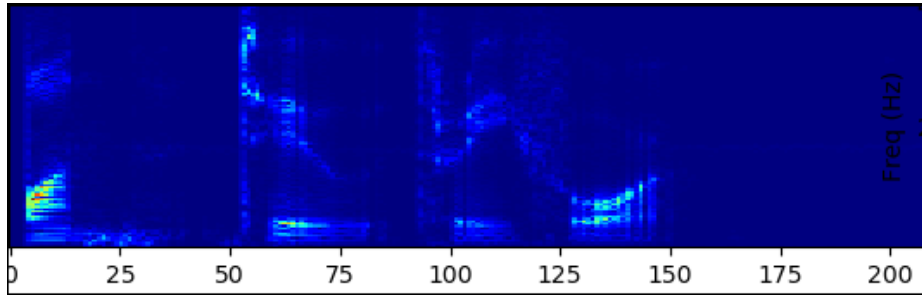


Fig. 15. Fourier Transform across all frames (Time →)

Power Spectrum

The power of the spectrum provides a more accurate representation of a human's audio perception. The following power transform is applied to the FT frequency spectrum.

$$P_k = \frac{|X_k|^2}{N}$$

where, X_k is the k th FFT frame.

The result is known as a Periodogram. The following illustration **Fig. 16** illustrates the power spectrum applied to **Fig. 15**

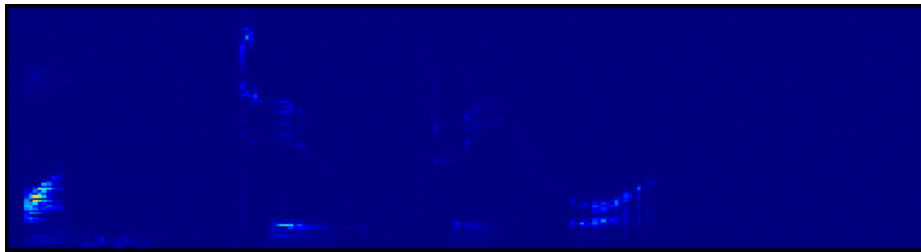


Fig. 16. Periodogram, power spectrum of all frames (Time →)

Mel Filter Bank

A series of mel-scale spaced triangular filters are applied to the Periodogram to remove excess information not perceived by the human ear.

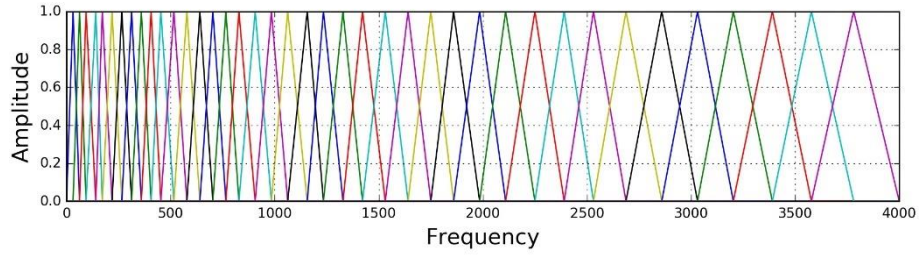


Fig. 17. Mel filter bank

A *Mel Filter Bank* is applied for two reasons. Firstly, humans do not differentiate between closely spaced frequencies. Because of this we can remove unnecessary information by applying a series of triangular frequency filters. And, secondly, we know that humans don't perceive frequencies linearly. The *mel-scale* is a 'frequency mapping' algorithm to adjust for this, hence the triangular filter banks are positioned and sized according to the mel-scale. Essentially the filter bank is mimicking perception of the human ear.

The equation below defines the filter bank where m is the number of filters we want, and $f(m)$ is the list of $m+2$ Mel-spaced frequencies. 40 is a common value chosen for the number of filter banks m .

$$Hm(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \leq k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

The following **Fig. 18** illustrates the Spectrogram; the result of applying the filter bank to the power spectrum (**Fig. 16**)

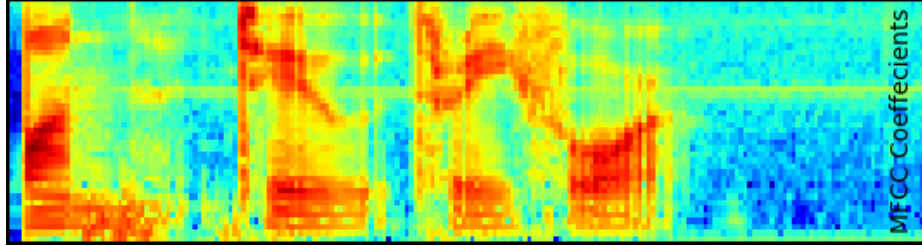


Fig. 18. Spectrogram, Mel filter banks applied to all frames (Time →)

Mel-frequency Cepstral Coefficients (MFCCs)

The final step is further compressing Spectrogram by application of a Discrete Cosine Transform (DCT). The result is an array of cepstral coefficients which will be included as feature vectors in the feature payload.

The DCT equation is shown below. Typically, 13 cepstral coefficients are calculated [0-12], but the first coefficient [0] is discarded as it provides no useful contribution to ASR.

$$C_i = \sum_{j=1}^{23} f_j \times \cos\left(\frac{\pi \times i}{23}(j - 0.5)\right), \quad 0 \leq i \leq 12$$

The following **Fig. 19** illustrates the complete sample of Mel-frequency Cepstral Coefficients, the result of applying the DCT to the Spectrogram (**Fig. 18**)

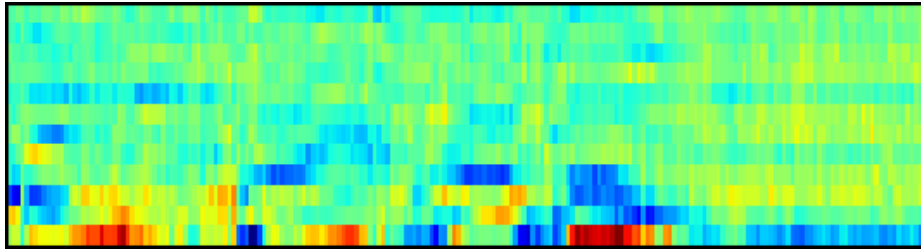


Fig. 19. Mel-frequency Cepstral Coefficients of all frames (Time →)

3.4 Server Pattern Matching & Decision Making

The REST API server uses speech feature vectors received from the client-side to *make a decision* on speaker authenticity.

The goal of the server is to compare the speech feature vector against a stored model, then return either a finite decision, or match likelihood. As discussed in the Literature Review, there are a range of approaches for matching and decision making (Page 11). Because the API's intention is speaker verification, which is often real-time, the speed of response is also an important consideration.

Algorithm Selection

The choice of server algorithm is does not significantly influence the research objectives, as the paper mainly wishes to demonstrate a REST API can successfully decouple the *feature extraction and selection* from *pattern matching and decision making*. In this context performance concerns mainly relate to client feature extraction, and any proven algorithm will suffice if similar benchmarks can be achieved for the distributed system. The *Gaussian Mixture Model* (GMM) is used, this has been used extensively in ASR systems and performs well for text-dependent speaker verification.

Gaussian Mixture Model (GMM)

A GMM works by expressing a feature vector in terms of a Gaussian distribution component contribution. This is easiest to understand in a one-dimensional space, as illustrated in **Fig. 20** (Note that for MFCC feature vectors normally consist of 12 coefficients, i.e. 12 dimensions).

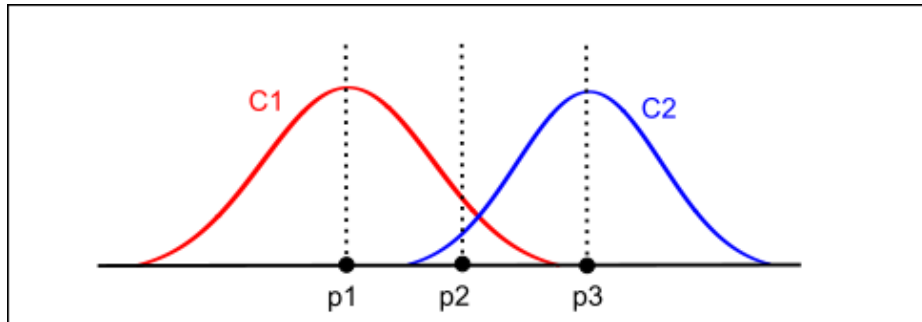


Fig. 20. One dimensional GMM explained

In **Fig. 20** each point, p_i , can be represented by probability of contributions from a set of gaussian components (C1 and C2). For example; the *first point*, p_1 , can be represented as $\{C1=1, C2=0\}$ (ie. The point p_1 is certainly described by C1, but not by C2). Similarly, the *third point*, p_3 , can be represented as $\{C1=0, C2=1\}$.

However, the *second point*, p_2 , best demonstrates how GMM works, p_2 is represented as a mixture of C1 and C2 (more specifically $\{C1=0.3, C2=0.2\}$). This can be interpreted as ‘the point p_2 is likely to be part of C2, but *a bit more likely* to be part of C1’. We’ll see how this is useful in *speaker modelling*.

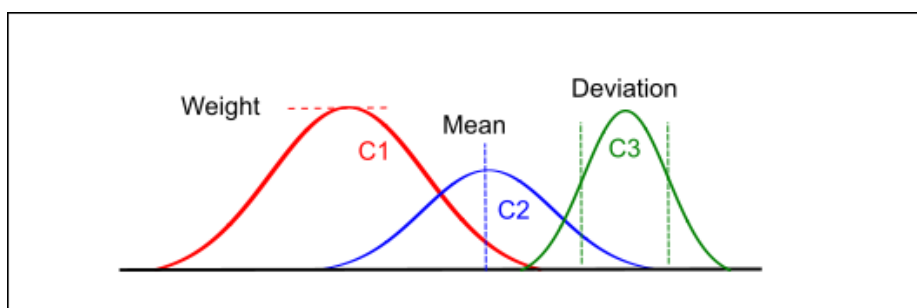


Fig. 21. GMM component parameters

The one-dimensional model (**Fig. 20**) simplifies the components for the purpose of illustration, only the *mean* of each distribution varied. A practical GMM is built with three component parameters; *mean*, *weight* and *variance* (variance being a product of deviation).

To relate the GMM to speaker verification we need to re-imagine the gaussian distributions in an n -dimensional space. The number of n -dimensions being dictated by the number of feature vectors chosen (in our case of 12 cepstral coefficients, ie. using a 12-dimensional GMM).

Therefore, the *speaker model* can be considered a collection of *gaussian components* that represent clusters of idiosyncratic speaker characteristics. The model parameters being defined by the *mean*, *weight* and *co-variance* matrices of each *gaussian component*. So each feature vector, when applied to *speaker model*, will result in a probability of it fitting (you may draw comparisons with how p_1 ‘fits’ to the ‘model’ in **Fig. 20**). The number of gaussian components used is dictated by trial, experience and performance requirements.

The speaker model is represented by the equation:

$$\lambda = \{p_i, \vec{\mu}_i, \Sigma_i\}$$

Where $i = 1, \dots, M$ components and p_i are the mixture weights, $\vec{\mu}_i$ are the mean vectors and Σ_i is the covariance matrix.

The probability of a feature vector \vec{x} being part of the model is defined by the following equation;

$$p(\vec{x}|\lambda) = \sum_{i=1}^M p_i b_i(\vec{x})$$

Where p_i are the mixture weights, and $b_i(\vec{x})$ the component densities. The component densities being defined as;

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{D/2} |\Sigma_i|^{1/2}} \exp \left\{ -\frac{1}{2} (\vec{x} - \vec{\mu}_i)^T \Sigma_i^{-1} (\vec{x} - \vec{\mu}_i) \right\}$$

3.5 Conclusion

This section expanded the theory and concepts underpinning a REST API approach to Distributed Speaker Verification. The following chapter details the approach to testing those theories and concepts.

4 Research Design

4.1 Introduction

In line with the research questions, the objective of the research paper is to demonstrate a successful distributed speaker verification REST API. The approach taken is production of two ASR systems:

- 1) A *laboratory reference system* which acts as a typical non-distributed ASR. The purpose of which provide a laboratory environment to test libraries, act as a reference and verify practical application of theory.
- 2) A *working web application* which utilises a REST API to demonstration a distributed ASR system. The purpose of which is to collect data for analysis and satisfy the research questions.

4.2 Design

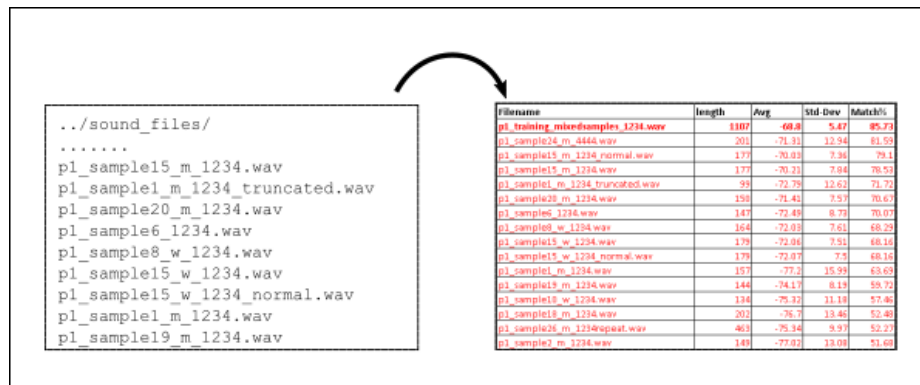
Laboratory ASR reference system

The purpose of a reference system is twofold: firstly, it provides a working reference model to benefit discovery and understanding; secondly, it provides a performance benchmark.

Implementation.

The system is coded in Python. Python was chosen because of its broad supports for mathematical, signal processing, and machine learning libraries. The system consists of two applications;

- **Rawsignal-example:** An application which processes a single pre-recorded input audio file (.wav). The application graphically plots each step of feature extraction, then compares the input file to a training file and outputs a text summary of results.
- **Batch-example:** An application which processes a batch of pre-recorded input audio files (.wav). The result is some summary text information plus a csv list (for spreadsheet import). Example output is shown in **Table 2**. Sample output from Laboratory Reference ASR., page 39.



The diagram illustrates the process of converting a folder of sound files into a performance spreadsheet data table. On the left, a box represents a folder named `../sound_files/` containing several .wav files. An arrow points from this folder to a table on the right, which represents the resulting performance data.

Filename	length	avg	Std Dev	Match%
p1_training_mixedsamples_1234.wav	1107	-68.8	5.47	85.73
p1_sample26_m_4844.wav	203	-71.31	12.34	61.52
p1_sample15_m_1234_normal.wav	177	-70.03	7.36	79.11
p1_sample15_m_1234.wav	177	-70.23	7.84	78.53
p1_sample15_m_1234_truncated.wav	99	-72.79	12.62	71.72
p1_sample28_m_1234.wav	150	-71.41	7.57	70.67
p1_sample6_1234.wav	147	-72.49	8.79	70.07
p1_sample15_w_1234.wav	164	-72.03	7.61	68.29
p1_sample15_w_1234.wav	179	-72.06	7.51	68.16
p1_sample15_w_1234_normal.wav	179	-72.07	7.5	68.16
p1_sample15_m_1234.wav	157	-77.2	15.99	63.69
p1_sample19_m_1234.wav	144	-74.17	8.19	58.32
p1_sample15_w_1234.wav	134	-75.32	11.18	57.46
p1_sample18_m_1234.wav	202	-76.7	13.46	52.46
p1_sample26_m_1234repeat.wav	403	-75.34	9.97	52.27
p1_sample7_m_1234.wav	149	-77.02	13.08	51.68

Fig. 22. Batch file converts folder of sound files to performance spread sheet data

The batch applications implement two options for feature extraction: the first perform the extraction in individual steps, using NumPy and SciPy Libraries (as per theory). The second strategy uses a library (python_speech_features) that provides built in methods for MFCC and LogEnergy feature extraction.

The applications use the machine learning library Scikit-learn GMM implementation.

Data Sources

In total 66 audio sound recording of the spoken digits ‘one, two, three, four’ were collected informally from 8 different people. 33 of the samples were recorded at varying time intervals by the author to be used as the ‘authorised speaker set’. The training data was constructed from an aggregated randomised sample of the ‘authorised speaker set’. The ‘authorised speaker set’ recordings were collected via phone (WhatsApp) and microphone. All other recordings were collected via phone (WhatsApp).

The recording sample file names were labelled in the following way ‘px_typey_source_1234_note.wav’.

Where;

- px - where x indicates the person (p1 is authorised speaker).
- typey - where *type* is either *sample* or *training* and y is an increment.
- *source* - is either m (microphone) or w (whatsapp).
- *1234* - indicates the spoken content.
- *note* – some additional comment such as normalised or truncated etc.

Samples used are listed in in **Table 2**. Sample output from Laboratory Reference ASR., p.39

Data Analysis and Processing/Scoring.

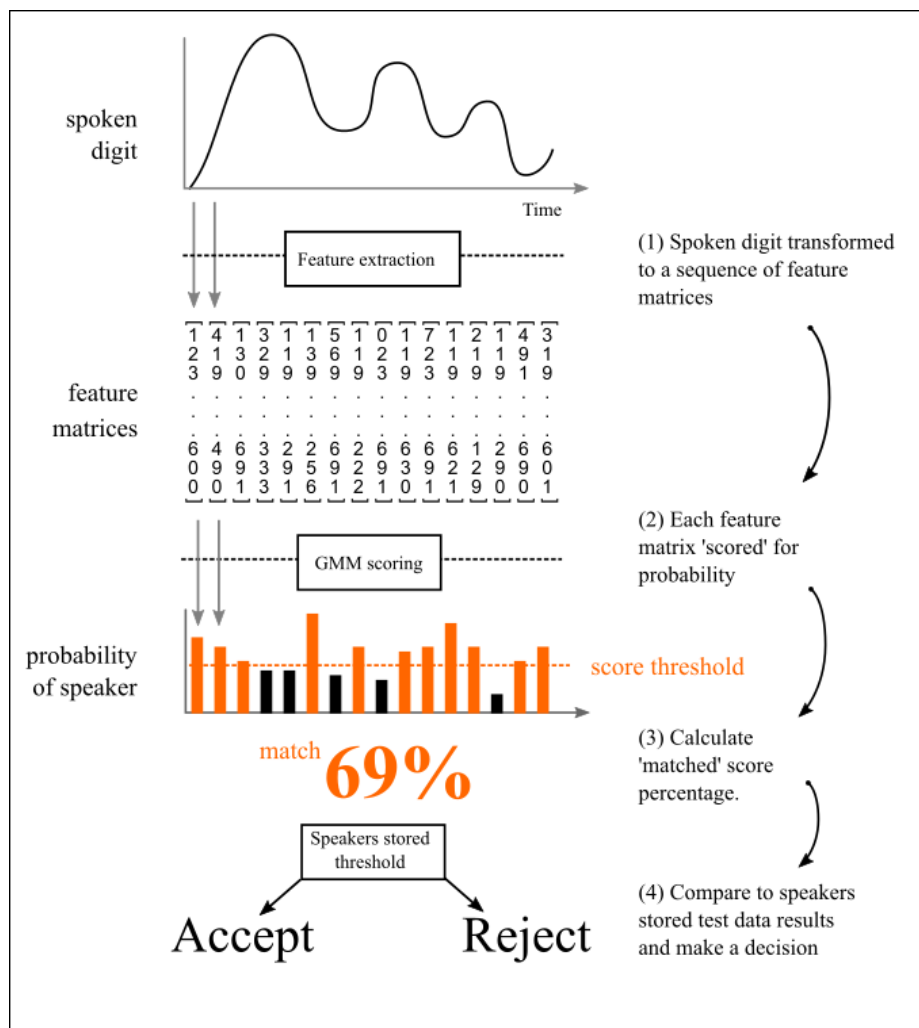


Fig. 23. Scoring and matching of spoken digits

An overall 'match' percentage is calculated for each processed sound file by counting the number of 'matched' frames. The higher the overall match percentage, the higher the probability of speaker authenticity. To see this illustrated refer to **Fig. 23**, above.

Note that there is no automatic *decision*, however previous test data can form the basis of a decision (Refer to *Benchmarking Performance*,p.30)

The first step in calculating an overall result is ‘scoring’ the frames (see **Framing**, p14). This scoring, results in a probability of each frame belonging to the speaker’s model. In more detail; every *frame* in the original sampled speech file is represented by a *feature vector* matrix. Each *feature vector* matrix is ‘scored’ by the *trained* GMM which results in an array of log probabilities, one numerical value for every frame. The next step determines if each frame belong to the speaker, to do this we need a log probability *score threshold* to compare with.

We calculate a ‘*score threshold*’ by scoring the speakers test data against the trained GMM, then averaging the resulting array of log probabilities less the resulting arrays standard deviation (note: this method of determining *score threshold* was derived from trial and error, it’s plausible that tuning the value would result in improved overall verification performance). The *score threshold* is a parameter stored per user and must be re-calculated any time the speaker model is re-trained.

The final step in calculating an overall result is to count the number of frames that match versus overall frames and publish a *match percentage*. To judge if a frame is a ‘*match*’ or ‘*not*’ each frames log probability is compared to the *score threshold*.

No automatic decision making is performed based on the overall *match percentage*. The resulting *match percentage* can be compared to a *match percentage* calculated from test data to give an informed decision or confidence. However, automated decision making requires further performance assessment. Refer to further discussion on benchmarking performance in the next section (p.30)

Source code and Publication

— The project files are publicly available on github – <https://github.com/footfish/python-speechprocessing-example>

**Batch data sound source not included*

Web Application - distributed ASR system utilising REST API.

The main objective of the research paper is to demonstrate a successful distributed speaker verification REST API. The web application is designed to collect research data, train speaker models, perform verification (scoring) and test results using the REST API. Further benchmarking to calculate equal-error-rate of application is performed with included command line tools.

Implementation.

The main application consists of a remote Python web server which serves a HTML/Javascript application file to the remote browser client. The browser running the HTML application communicates with the remote server via a REST API. For deployment Heroku (cloud application delivery service) is used. Additional command line scripts are also provided to examine and benchmark performance.

The voice *feature extraction* takes place live in the browser, making use of Web Audio API (Web Audio API n.d.) and Meyda (Meyda, Audio feature extraction for JavaScript. n.d.) Javascript audio feature extraction library. The Python server makes use of the same libraries and similar methods to the reference application. The lightweight Python web application framework, Flask, is used.

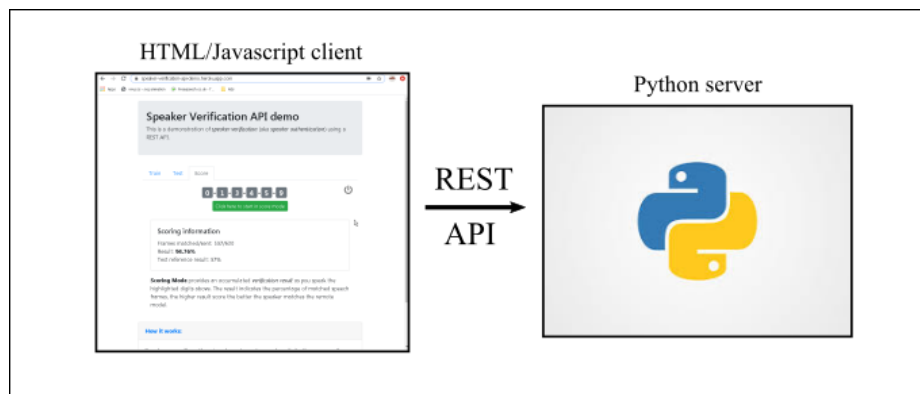


Fig. 24. Client-Server

Main application features:

- User registration form
- Session login and persistence
- Live training of speaker models
- Live verification scoring
- Training and scoring log (with speech feature data)
- Intrusion testing (score other users against own speaker model)

- Command line tools for further analysis (equal error rate)
- REST API supports 8 calls:
 - **doc**: provides basic API information
 - **user**: registers a user
 - **session**: Logs in a user
 - **delete**: Clears the trained model and test data
 - **status**: Returns the size of trained model and test data
 - **train**: Sends feature vectors for training and testing, returns result
 - **intrusionTest**: Tests speakers' model against intruders (other users), returns results
 - **score**: Sends feature vectors for scoring and returns result

Data Analysis and Processing/Scoring

The speaker verification application works by collecting training, testing and scoring data in a live web session. The API server (remote) fundamentals of scoring and training are the same as those previously detailed in the laboratory ASR system (refer to p23).

The application is trained by speaking the digits 0-9 several times (digits are randomly re-ordered). Users are prompted to speak the highlighted digit out loud in sequence, refer to the screenshot **Fig. 25** of training in progress below. The trained digits are alternatively used for a train and data set ('training set' trains the model, 'test set' fixes score thresholds). All the training data is logged.

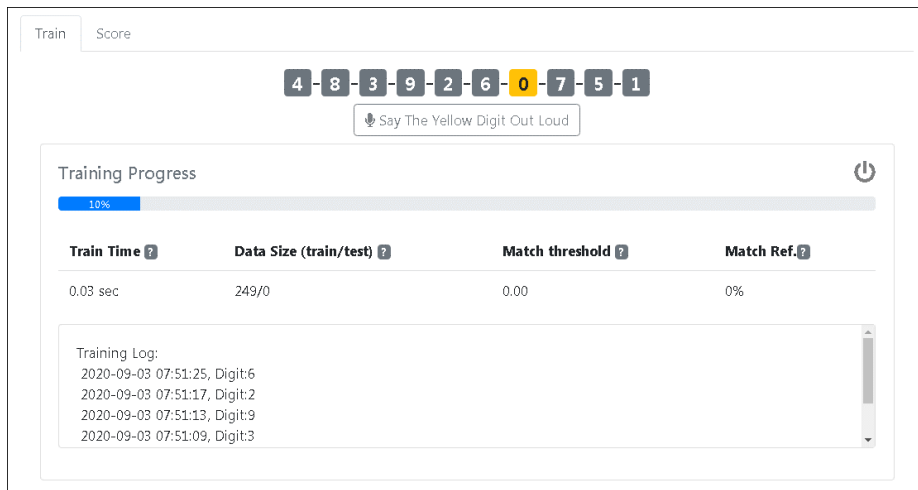


Fig. 25. Live web 'model training', yellow digit prompts speaking of '0'

Once the model has been trained an *intrusion test* may be performed. The *intrusion test* is performed by scoring imposters' (other users) test data against the trained model. See screenshot of the intrusion test **Fig. 26**.

Intrusion Test		
Intruder	Match	Data Size
DeletePlease(you!)	90%	891
wavedavey	10%	806
Hilts	22%	439
Nola_07	72%	584
TheMaster	94%	595
MikeK	14%	1069
BenOS	57%	1148
Michael	38%	492
JakeM	80%	620
Trer_71	74%	600
Avinash	47%	672

Fig. 26. Intruder test scores 'imposters' speech data against your trained model.

When the application is fully trained scoring may be performed. For scoring, a sequence of 5 random digits are presented. Users are prompted to speak the highlighted digit out loud in sequence, refer to the screenshot **Fig. 27** of scoring in progress below. An ongoing tally of match results are presented as the digits are read aloud. The final 'match result' is presented as the percentage of matched frames for all five digits. All digits read aloud are logged in a 'score store'.

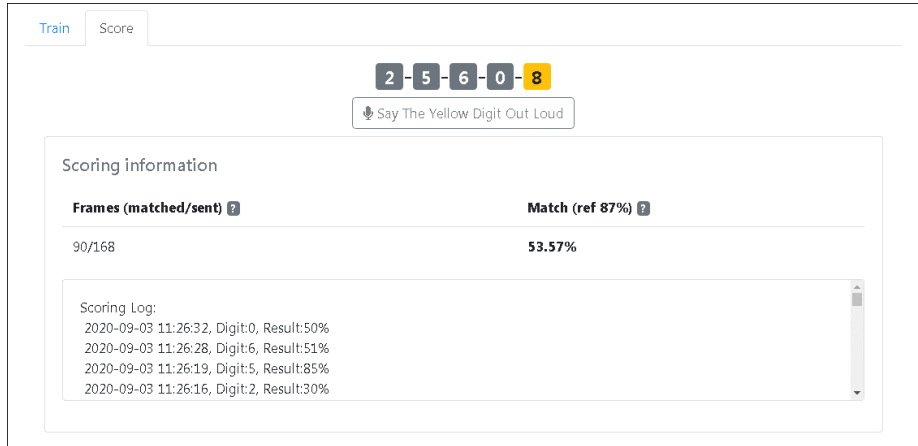


Fig. 27. Live web ‘verification scoring’, results of previous scoring shown in log.

The application currently allows for informal scoring as illustrated above in **Fig. 27**. The scoring mechanism is described in the reference application *Data Analysis and Processing/Scoring*.p.25.

Benchmarking Performance

Because system benchmarking is a processor intensive task, an ‘offline’ command line tool is used. The benchmark used is the *Equal Error Rate (EER)*.

A common interpretation of ASR performance is determined with the equal-error-rate (Hansen and Hasan 2015). The *Equal Error Rate (EER)* is found by adjusting the speakers *decision threshold* (not to be confused with *score threshold* – see **Fig. 23**) until false accepts match false rejects. A lower *EER* indicates better performance (i.e. less false rejects, less false accepts).

The *EER* calculation requires recording of two types of error which normally occur in speaker verification;

- A false **accept** (FA), where an **imposter** is allowed.
- A false **reject** (FR), where a **legitimate** speaker is denied (a false accept is generally considered worse than a false reject).

Logging of these two metrics of interest over time facilitates determination of the corresponding error rates: *False Acceptance Rate (FAR)* and *False Rejection Rate (FRR)*. The *Equal Error Rate (EER)* may then be found by adjusting the threshold until *FAR* and *FRR* are equal. This is illustrated graphically in **Fig. 28** below.

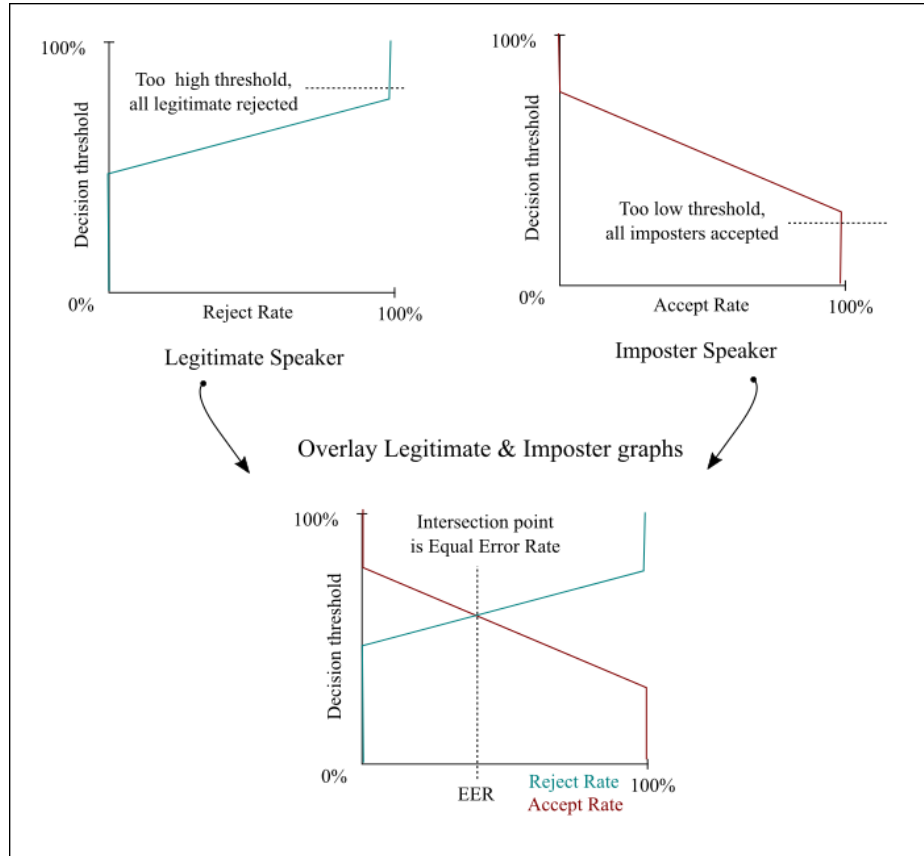


Fig. 28. Determining EER (Equal Error Rate).

To facilitate EER calculation logging of features extracted from spoken digits is required. Furthermore, both *imposter* and *legitimate speaker* is needed. The application logging feature includes all trained and scored digits in two separate stores (the training store containing *test* and *train* data). In the context of this application the ‘logged-in’ user is treated as the *legitimate speaker* and the ‘other’ users are treated as imposters. The data source used is listed in the following section **Table 1**. List of data source contributors.

The process used for determination of *Equal Error Rate* is as follows:

- (1) Generate a list of all valid users (filter users with poor training data).
- (2) Iterate through the user list treating each as a *legitimate speaker*.
- (3) Load all the logged digits for the *legitimate speaker* excluding training digits (i.e. test and score log).

- (4) Divide the logged digits into 3-digit sets ('3' is parameterised and can be changed).
- (5) Iterate through the legitimate 3-digit sets calculating the *match result percentage* (**Fig. 29**).
- (6) For each iteration pick an arbitrary imposter and an arbitrary 3-digit set and calculating the *match result percentage* (**Fig. 29**).
- (7) The iterative result is an array of imposter/legitimate paired *match results*. For each of these pairs an EER value may be calculated by iteratively checking the FRR and FAR until equal (**Fig. 29**).
- (8) Multiple runs may be performed to improve accuracy.

```
User: 'wavedavey' digits recorded: 70

Usr: wavedavey Digits:[2, 8, 6] avg:-33.5 match:100.0%
Imp: TeriP Digits:[2, 9, 5] avg:-241.79 match:10.34%
Usr: wavedavey Digits:[9, 5, 1] avg:-33.04 match:100.0%
Imp: David Digits:[4, 5, 4] avg:-45.96 match:43.48%
. . . . .
. . . . .
. . . . .
. . . . .
Usr: wavedavey Digits:[1, 0, 8] avg:-35.91 match:90.91%
Imp: Sean1 Digits:[3, 7, 2] avg:-85.36 match:0.0%
Usr: wavedavey Digits:[1, 0, 6] avg:-33.1 match:100.0%
Imp: JakeM Digits:[0, 9, 5] avg:-53.61 match:29.17%

False Accept Rate (FAR):4.35% False Reject Rate (FRR):4.35% Match
Threshold:81.7%
```

Fig. 29. Example of EER calculation (legitimate speaker 'wavedavey')

Data Sources.

The web application is configured for recording and collection of spoken data (through the API). The data collected is stored as *feature vectors* with associated user information, session information and a digit label (which digit was spoken). All contributors accepted the data collected to be provided and stored as open data under Creative Commons Attribution.

The test population was obtained by distribution of an application registration link via the authors social contacts. The data was collected on the contributors own phone, tablet or personal computer. The sample database is included with the github project files (*Source code and Publication p34*).

The following contributor data is collected and stored:

- 'User credential' data (email/username) * NOTE: This is used for session login and not exported with sample database.

- 'Alias' data collected during registration.
- 'Speech feature vector' data.
- 'Demographic' data collected on the form during registration (gender, age group).
- 'Performance log' data collected and calculated during speech training and scoring.
- 'Browser information' data collected from browser headers during session login.

In total there are 41 contributors, 11 Female, 30 Male with 3637 digits recorded. The table below lists the contributors. The digit stored consist of those used for training & testing (train store) and those used for random score tests.

Speaker			Digit store	
alias	gender	age	train	score
wavedavey	Male	40-49	60	40
Hilts	Male	50-59	40	60
Nola_07	Female	Prefer Not to say	50	0
TheMaster	Male	40-49	40	60
CoolDude	Male	16-19	60	60
MikeK	Male	40-49	60	15
BenOS	Male	50-59	60	25
Michael	Male	70-79	60	60
JakeM	Male	16-19	60	60
Trer_71	Male	40-49	60	15
Avinash	Male	40-49	60	60
Duncan	Male	40-49	40	0
Niall	Male	50-59	50	5
Sean1	Male	40-49	60	0
Ronan B	Male	40-49	60	30
mrangry	Male	50-59	50	20
Elizabeth W.	Female	50-59	50	25
hivemindx	Male	50-59	40	25
TaraM	Female	40-49	40	10
TerryD	Male	50-59	60	60
Padraig	Male	50-59	40	25
j_mcg	Male	50-59	50	35
Seven Whistler	Male	60-69	42	10
Denis	Male	40-49	60	60

Maria	Female	50-59	60	60
David	Male	70-79	60	30
PaulMcC	Male	40-49	60	60
12345	Male	50-59	60	0
Fri123	Female	40-49	60	35
fergie	Male	40-49	60	0
OwenP	Male	50-59	60	20
Fintan	Male	40-49	60	60
TeriP	Female	40-49	60	15
FinoH	Female	50-59	60	30
Robert	Male	40-49	60	40
MayaB	Female	20-29	60	30
gwenreil	Female	40-49	60	20
Macwfd	Female	50-59	60	40
Djfdemo	Male	40-49	60	45
ThumbsUp	Male	16-19	60	60
IDoNotCare	Female	40-49	60	60

Table 1. List of data source contributors

Source code and Publication

- The project files are available on github – <https://github.com/footfish/speaker-verification-api-demo>
- Live application - <https://speaker-verification-api-demo.herokuapp.com/>

The following chapter details the presentation of findings.

5 Presentation of Findings

Two applications are presented. The first application, a ‘*Laboratory ASR reference system*’, is designed to demonstrate working speaker validation without presenting formalised findings.

The second application, a ‘*Web Application - distributed ASR system utilising REST API*’ is designed to formally address the research objective (1.3 page 2) and answer the posed research questions (1.4 page 3).

The following is a summary of findings from each application.

Laboratory ASR reference system

Application rawsignal-example:

The application rawsignal-example outputs three things. **Fig. 30**

- i) A sequence of plots illustrating each step in feature extraction for a single frames. **Fig. 31**
- ii) A sequence of plots illustrating each step in feature extraction for the whole signal (all frames). **Fig. 32**
- iii) A summary of the feature extraction and GMM test scoring.

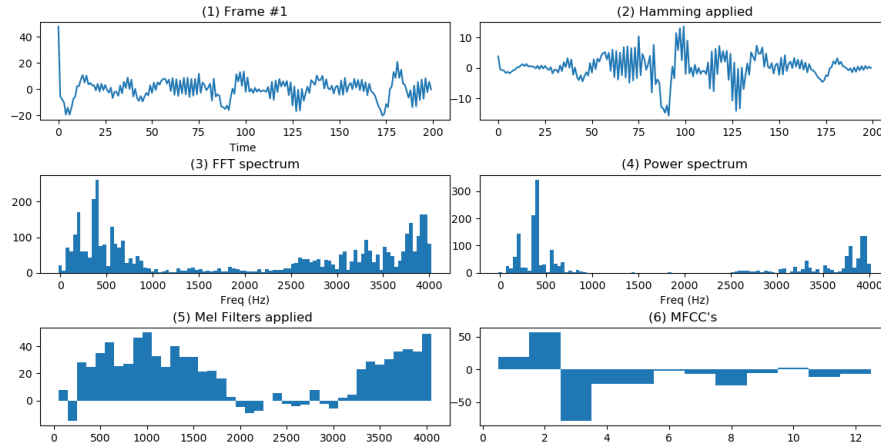


Fig. 30. Application Rawsignal-example: Feature extraction steps for a single frame

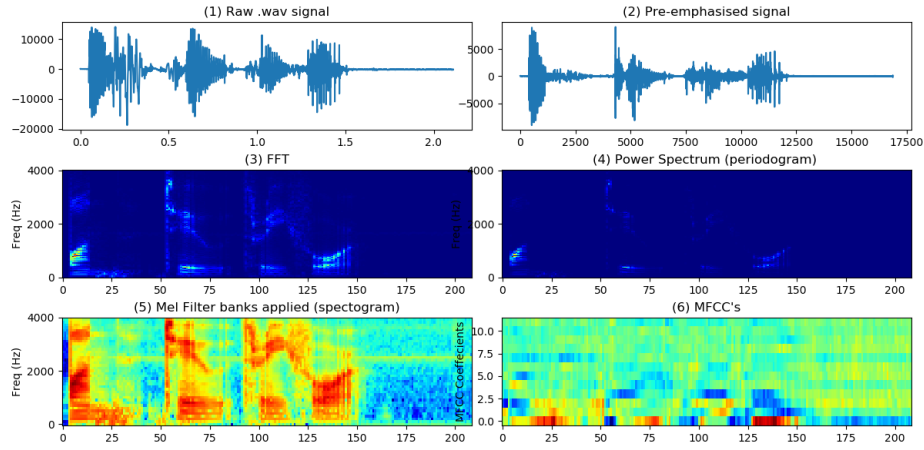


Fig. 31. Application Rawsignal-example: Feature extraction steps for complete signal

```
processing-example/src/rawsignal-example.py"
Sound file duration 2.695375(secs), 21563 samples
Sampling rate 8000Hz (8000 samples per sec)
Min/max amplitude: -10000.0/8445.0
-Framing-
Frame length: 200 samples, 25ms
Frame overlap: 15ms
No. of frames: 268
-Filter Banks-
No. of filter banks: 40 (spectrogram array size)
-MCC's-
No. of coeffecients: 12
Min/Max coeffecient: -173.8211009964551/101.51229021673885
Min/Max coeffecient(2): -81.82036708104546/35.85068591955849
MFCC length: 268
-GMM-
Average: -47.156211495638644
Deviation: 3.4385192518357854
Min/Max value: -57.527097353600325/-40.2975258186964
Result length: 268
```

Fig. 32. Summary text output from a processed sound file.

Application batch-example:.

The application batch-example outputs a CSV table of results. The following **Error! Reference source not found.** illustrates results sorted according to match score (sorting and coloring is performed in spread sheet). The 'p1' in the Filename column indicates

the file was recorded by the author (speaker who should be authorised), for convenience these are highlighted in red. The expectation is that ‘p1’ is grouped at the top.

The following are field columns:

- Filename: name of file being processed (interpretation refer to *Data Sources*, p24).
- Length: number of frames.
- Avg: The log probability GMM array result average.
- Std-Dev: The log probability GMM array result standard deviation.
- Match %. The % number of frames considered to ‘match’ the GMM (higher is better).

Note that the first sound file is the training data. This is a ‘sample’ set and many different parameter settings and combinations were performed.

Filename	length	Avg	Std-Dev	Match%
p1_training_mixedsamples_1234.wav	1107	-68.8	5.47	85.73
p1_sample24_m_4444.wav	201	-71.31	12.94	81.59
p1_sample15_m_1234_normal.wav	177	-70.03	7.36	79.1
p1_sample15_m_1234.wav	177	-70.21	7.84	78.53
p1_sample1_m_1234_truncated.wav	99	-72.79	12.62	71.72
p1_sample20_m_1234.wav	150	-71.41	7.57	70.67
p1_sample6_1234.wav	147	-72.49	8.73	70.07
p1_sample8_w_1234.wav	164	-72.03	7.61	68.29
p1_sample15_w_1234.wav	179	-72.06	7.51	68.16
p1_sample15_w_1234_normal.wav	179	-72.07	7.5	68.16
p1_sample1_m_1234.wav	157	-77.2	15.99	63.69
p1_sample19_m_1234.wav	144	-74.17	8.19	59.72
p1_sample10_w_1234.wav	134	-75.32	11.18	57.46
p1_sample18_m_1234.wav	202	-76.7	13.46	52.48
p1_sample26_m_1234repeat.wav	463	-75.34	9.97	52.27
p1_sample2_m_1234.wav	149	-77.02	13.08	51.68
p1_sample21_m_1111.wav	173	-77.55	10.74	50.29
p1_sample17_m_1234.wav	146	-77.79	12.29	49.32
p1_sample26_w_1234repeat.wav	466	-76.07	9.9	48.5
p1_sample13_w_1234.wav	161	-75.27	7.83	47.2

p1_training_samples1-15_1234.wav	1885	-78.59	13.5	47.11
p1_sample9_w_1234.wav	137	-77.78	11.81	46.72
p1_sample23_m_3333.wav	166	-77.92	11.92	41.57
p4_sample1_w_1234.wav	150	-79.59	13.58	36.67
p1_training_1234.wav	661	-79.54	11.44	36.31
p1_sample16_m_1234.wav	160	-81.91	13.48	36.25
p1_sample7_w_1234.wav	169	-80.57	12.71	34.91
p1_sample22_m_2222.wav	140	-80.35	12.21	32.86
p4_sample4_w_1234.wav	149	-81.5	12.54	31.54
p1_sample3_m_1234.wav	171	-87.04	17.42	30.41
p1_sample5_m_1234.wav	150	-83.91	15.78	29.33
p1_sample4_m_1234.wav	152	-86.82	19.12	26.97
p8_sample2_w_1234.wav	186	-84	13.55	26.34
p1_sample12_w_1234.wav	134	-83.64	13.38	26.12
p1_sample11_w_1234.wav	138	-81.83	12.13	26.09
p1_sample14_w_1234.wav	103	-85.11	13.55	20.39
p4_sample2_w_1234.wav	154	-82.06	10.95	20.13
p3_sample2_w_1234.wav	413	-84.35	11.51	18.89
p8_sample1_w_1234.wav	196	-89.9	18.76	18.37
p4_sample3_w_1234.wav	164	-82.37	10.89	18.29
p7_sample4_w_1234.wav	122	-86.12	13.39	14.75
p3_sample3_w_1234.wav	246	-88.84	14.91	14.23
p7_sample3_w_1234.wav	145	-87.05	14.67	13.79
p6_sample2_w_1234.wav	212	-90.01	15.48	13.21
p6_sample6_w_1234.wav	141	-90.42	14.51	12.77
p7_sample1_w_1234.wav	160	-87.46	15.31	12.5
p6_sample4_w_1234.wav	226	-87.31	12.38	12.39
p8_sample3_w_1234.wav	187	-91.41	15.89	11.76
p6_sample3_w_1234.wav	156	-91.51	15.02	10.9
p3_sample4_w_1234.wav	248	-86.36	9.93	10.89
p5_sample2_w_1234.wav	260	-87.91	10.83	10.77
p2_sample1_w_1234.wav	205	-89.72	12.56	10.73
p2_sample5_w_1234.wav	198	-88.06	11.94	10.61
p7_sample2_w_1234.wav	176	-89.33	14.06	10.23

p2_sample6_w_1234.wav	288	-84.35	10.02	7.99
p6_sample1_w_1234.wav	165	-91.67	11.55	7.88
p5_sample3_w_1234.wav	218	-94.57	15.52	7.34
p6_sample5_w_1234.wav	114	-91.78	13.89	7.02
p2_sample2_w_1234.wav	175	-87.83	10.71	6.86
p2_sample7_w_1234.wav	284	-86.93	9.13	4.93
p3_sample1_w_1234.wav	217	-93.79	16.3	4.15
p2_sample3_w_1234.wav	145	-88.36	10.27	4.14
p5_sample1_w_1234.wav	226	-91.23	10.9	3.98
p2_sample8_w_1234.wav	224	-87.4	9.57	3.57
p5_sample4_w_1234.wav	142	-99.03	15.08	3.52
p2_sample4_w_1234.wav	137	-91.79	11.68	2.92
p1_sample25_m_1234_acting.wav	159	-96.11	13.41	1.89

Table 2. Sample output from Laboratory Reference ASR.

Web Application - distributed ASR system utilising REST API.

The web application is used to collect, train, test and score speaker data during a live web session. Due to the intensive nature of benchmarking (recursive iterations) a command line tool is used to perform benchmarking.

Scoring information	
Frames (matched/sent) ?	Match (ref 87%) ?
97/180	53.89%

Fig. 33. Live scoring information presented in the Web Application

Web application – stored results:

Each speaker contributor completed a registration form (*Data Sources.p32*) and performed training and scoring on their own device. During the training phase, *test data* is used to calculate a *reference match %*, a high *match %* indicates valid speaker recognition has been performed (ref *Data Analysis and Processing/Scoring.p25*). The logged results are summarised in **Table 3** below.

Speaker			Digit store		Results	
alias	gender	age	train	score	Match %	Threshold
wavedavey	Male	40-49	60	40	95	-46.29
Hilts	Male	50-59	40	60	94	-41.82
Nola_07	Female	Prefer Not to say	50	0	98	-43.25
TheMaster	Male	40-49	40	60	87	-32.73
CoolDude	Male	16-19	60	60	91	-39.56
MikeK	Male	40-49	60	15	85	-38.56
BenOS	Male	50-59	60	25	88	-41.12
Michael	Male	70-79	60	60	90	-39.75
JakeM	Male	16-19	60	60	94	-33.55
Trer_71	Male	40-49	60	15	88	-34.15
Avinash	Male	40-49	40	45	93	-45.33
Duncan	Male	40-49	40	0	89	-35.99
Niall	Male	50-59	50	5	97	-232.53
Sean1	Male	40-49	60	0	90	-39.42
Ronan B	Male	40-49	60	30	95	-35.28
mrangry	Male	50-59	50	20	89	-39.31
Elizabeth W.	Female	50-59	50	25	89	-44.21
hivemindx	Male	50-59	40	25	98	-527.86
TaraM	Female	40-49	40	10	100	-78.57
TerryD	Male	50-59	60	60	85	-35.12
Padraig	Male	50-59	40	25	95	-144.13
j_mcg	Male	50-59	50	35	95	-63.00
Seven Whistler	Male	60-69	42	10	90	-37.86
Denis	Male	40-49	60	60	97	-44.25
Maria	Female	50-59	60	60	89	-45.30
David	Male	70-79	60	30	82	-35.93
PaulMcC	Male	40-49	60	60	95	-39.44
12345	Male	50-59	60	0	98	-46.46
Fri123	Female	40-49	60	35	99	-60.99
fergie	Male	40-49	60	0	94	-46.40
OwenP	Male	50-59	60	20	97	-107.79

Fintan	Male	40-49	60	60	90	-35.92
TeriP	Female	40-49	60	15	95	-57.57
FinoH	Female	50-59	60	30	91	-37.55
Robert	Male	40-49	60	40	88	-33.85
MayaB	Female	20-29	60	30	92	-37.97
gwenreil	Female	40-49	60	20	90	-36.09
Macwfd	Female	50-59	60	40	94	-40.97
Djfdemo	Male	40-49	60	45	97	-67.55
ThumbsUp	Male	16-19	60	60	86	-33.41
IDoNotCare	Female	40-49	60	60	91	-46.20

Table 3. Contributors with summary results (command line output *userList.py*).

During the training phase, *test data* is also used to calculate an individual's *score Threshold*. The *score Threshold* is a log probability value used to determine if individual speech frames are counted (ref *Data Analysis and Processing/Scoring*.p25). A very low *score Threshold* indicates the *speaker model* is not well trained (i.e. any arbitrary speech frames will be counted). This problem is generally solved by more training, or retraining. For the purposes of benchmarking, contributors with *score Threshold* less than -70 are excluded (those highlighted grey in **Table 3** above.).

Equal-Error-Rate, command line benchmarking.

A common interpretation of ASR performance is determined with the Equal Error Rate, where false acceptance rate (FAR) and false rejection rate (FRR) are equal. (for description see *Benchmarking Performance* p30).

Analysis results for three passes of the command line tool for a **population of 35** contributors results in an overall **average EER of 8.9%**, using **3-digit** sets. Results are tabulated below, all values in percentages.

Table legend: FAR – False Acceptance Rate, FRR – False Rejection Rate, eerRef – Equal Error Rate match %, usrRef – Users stored match % (calculated with test data).

	Pass 1			Pass 2			Pass 3			Test data
Alias	FAR %	FRR %	eerRef %	FAR %	FRR %	eerRef %	FAR %	FRR %	eerRef %	usrRef %
wavedavey	2.17	2.17	66.5	2.17	2.17	49.8	2.17	2.17	65.1	95
Hilts	10	10	58	10	10	58.2	12	12	63.5	94
Nola_07	8.33	8.33	85.6	8.33	8.33	97.1	0	0	84.1	98
TheMaster	5	5	54.4	7.5	7.5	72.3	5	5	66.5	87
MikeK	13.33	13.33	73.2	6.67	6.67	70.7	13.33	13.33	73.2	85
BenOS	8.33	8.33	84.8	5.56	5.56	81.9	2.78	2.78	77.8	88
Michael	10	10	84.1	11.67	11.67	86.8	15	15	91.1	90
JakeM	3.33	3.33	74.8	5	5	76.8	3.33	3.33	72.6	94
Trer_71	16.67	16.67	82.6	6.67	6.67	77.6	13.33	6.67	77.6	88
Avinash	13.33	13.33	69.8	13.33	13.33	69.8	20	18.33	74.8	85
Duncan	8.33	8.33	90.8	0	0	62.3	8.33	8.33	99.8	89
Sean1	12.5	6.25	99.8	6.25	6.25	94.8	6.25	6.25	99.8	90
Ronan B	7.5	7.5	83.2	5	5	82.6	2.5	2.5	79.8	95
mrangry	4.17	4.17	86.5	8.33	8.33	87.3	4.17	4.17	79.8	89
Elizabeth W.	7.69	7.69	74.8	7.69	7.69	61.7	11.54	11.54	77.3	89
TerryD	10.34	10.34	76.8	5.17	5.17	71.7	12.07	12.07	78.6	85
j_mcg	11.11	11.11	94.3	11.11	11.11	95.5	5.56	5.56	87.3	95
Seven Whistler	15	15	85.3	5	5	79.8	5	5	79.8	90
Denis	5.36	5.36	72.6	8.93	8.93	82.5	8.93	8.93	81.4	97
Maria	16.67	16.67	83.2	18.33	18.33	83.8	18.33	16.67	83.2	89
David	18.42	18.42	55.8	21.05	21.05	57	18.42	18.42	55.8	82
PaulMcC	3.33	3.33	83.2	3.33	3.33	83.2	6.67	6.67	86.5	95
12345	5	5	94.8	10	10	99.8	10	10	99.8	98
Fri123	20	7.5	99.8	15	7.5	99.8	25	7.5	99.8	99
fergie	5	5	84.5	10	10	99.8	15	10	93.6	94
Fintan	1.79	1.79	47.7	0	0	44.8	1.79	1.79	47.7	90
TeriP	16.67	13.33	99.8	23.33	13.33	99.8	20	13.33	99.8	95
FinoH	5	5	53.7	15	15	71.3	7.5	7.5	55.4	91

Robert	9.09	6.82	74.8	9.09	6.82	74.8	13.64	13.64	78.6	88
MayaB	12.5	12.5	79	5	5	66.5	7.5	5	54.8	92
gwenreil	6.25	6.25	69.1	6.25	6.25	75.6	3.12	3.12	59.8	90
Macwfd	13.04	13.04	86.5	19.57	17.39	89.8	8.7	8.7	80.9	94
Djfdemo	12	0	99.8	10	0	99.8	14	0	99.8	97
ThumbsUp	6.67	6.67	67.6	1.67	1.67	64.8	5	5	66.5	86
IDoNotCare	15.52	15.52	91.5	13.79	13.79	90.8	15.52	15.52	93.2	91
Average rates %	9.7	8.66		9.02	8.11		9.76	8.17		
Standard deviation	4.86	4.63		5.59	5.04		6.11	5.09		

Table 4. EER results: overall *average EER of 8.9%*, using **3-digit** sets

To give comparative context to overall average EER 8.9% for a 3-digit set we may refer to **Table 5** below which is an extract from Joseph Campbell's '*Speaker Recognition: A Tutorial*' (Campbell 1997) results.

	Input	Text	Pop	EER
Tisby 1991	Telephone	isolated digits	100	2.8% (1.5s) 0.8% (3.5s)
Reynolds 1995	Office	Dependent	138	0.12% (10s)
Colombi et al 1996	Office	Dependent	138	0.28% (10s)
Reynolds 1996	Telephone (mismatched handsets)	Independent	416	11-16% (3s) 6-8% (10s)

Table 5. EER rates extract Joseph Campbell's '*Speaker Recognition: A Tutorial*' (Campbell 1997)

Observations

The EER estimation produced fair results considering the nature of data collection (contributors' own device). However, there several observations associated with the results:

- Some contributors exhibited difficulty in making recordings. This was typically a particular device, or trouble with a particular digit. The possible causes are a mix of: channel effects, device compatibility or the primitive implementation of 'silence detection' on the browser. The result of this may be some poor-quality data.
- No feature in the application was included to allow deletion or removal of false recordings in the digit set. The result of this may be some poor-quality data.

- Some contributors had particularly poor performance. This could possibly be due to the two points above but requires further investigation.
- Recordings were generally gathered in a single session. The result is no little variance in the contributed speakers' voice samples.
- Some training score *match % thresholds* are very high (close to 100%), this indicates the model is unlikely to perform well in a real-world application.
- The overall system has many parameter variables, on the client and server. It is likely verification performance could be improved, if time was given to parameter optimisation.

6 Conclusion

A considerable amount of work has been completed to build and present the REST API application. The overall research conclusions should be addressed in terms of the research questions raised.

Research Question 1 - Conclusion

Can a distributed speaker verification REST API, which decouples feature extraction and selection from feature vector matching and decision making, be used to verify speakers using a short sequence of spoken numeric digits?

The system demonstrated a working model with reasonable performance over a short sequence of digits. However, a real-world solution poses many challenges not addressed in this study.

Research Question 2 - Conclusion

Can a regular web browser client successfully perform real-time speech feature extraction and selection for a distributed speaker verification REST API?

The system demonstrated a generic web browser can perform real-time feature extraction and selection successfully. However, some specific browsers and devices had problems which require further study.

Research Question 3 - Conclusion

What is the real-world verification performance of the distributed speaker verification REST API?

The system demonstrated an overall average EER 8.9% in what could be considered real-world environment. However, the population used for analysis was small and average EER standard deviation was high at 5.22. An EER of 8.9% could be considered acceptable in certain applications, for example where an increased confidence score is required, such as credit card authorisation.

Summary

The research demonstrated the feasibility of using a RESTful approach to speaker verification. The research provided a corpus of data, and an innovative method of data collection, which may be useful in further studies. The research applications developed are prototypes, leaving much scope for further investigation and refinement.

Glossary of Terms

- **REST** – Representational State Transfer. A stateless architectural software design style often used for Web Services. Credited for being fast, reliable and scalable.
- **API** – Application Interface. A defined software interface.
- **ASR** – Automatic Speaker Recognition. Deciding ‘who’ is speaking without human intervention (p.8).
- **ASV** – Automatic Speaker Verification. A type of ASR which focusses on authentication.
- **GMM** – Gaussian Mixture Model. A probabilistic mathematic model, in the context of this paper it’s applied to modelling the ‘speaker’ (p.20).
- **MFCC** - Mel-frequency cepstral coefficient. A group of coefficients used to represent the short-term power spectrum of a sound (p.11, p.19).
- **DFT** - Discrete Fourier Transform. A discrete algorithm for mapping sound from time to frequency domain (p.10)

Bibliography

- Atal, B S. 1974. "Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification." *The Journal of the Acoustical Society of America* 55. doi:10.1121/1.1914702.
- Baker, J., Li Deng, James Glass, Sanjeev Khudanpur, Chin-Hul Lee, Nelson Morgan, and Douglas O'Shughnessy. 2009. "Research Developments and Directions in Speech Recognition and Understanding, Part 1." *IEEE Signal Processing Magazine* 26. <https://microsoft.com/en-us/research/publication/research-developments-and-directions-in-speech-recognition-and-understanding-part-1>.
- Beigi, Homayoon. 2011. *Fundamentals of Speaker Recognition*. Springer. <http://www.fundamentalsofspeakerrecognition.org>.
- . 2010. *Speaker Recognition – Practical Issues. (A Tutorial)*. http://www.recotechnologies.com/~beigi/ps/speechtek2010_homayoon_beigi.pdf.
- . 2012. *Speaker Recognition: Advancements and Challenges*. Vol. New Trends and Developments in Biometrics. INTECH. doi:10.5772/52023.
- Benesty, Jacob, M. M. Sondhi, and Yiteng Huang. n.d. *Springer Handbook of Speech Processing*. Springer Science & Business Media.
- Broun, Charles C., William M. Campbell, David Pearce, and Holly Kelleher. 2002. "Distributed Speaker Recognition Using the ETSI Distributed Speech Recognition Standard."
- Campbell, Joseph (jr) P. 1997. *Speaker Recognition: A Tutorial*. Vols. PROCEEDINGS OF THE IEEE, VOL. 85, NO. 9. IEEE. <https://pdfs.semanticscholar.org/298c/d5cefd80cd2aa0e9bc1d27f552b9eb18633.pdf>.
- Caridakis, George, Ginevra Castellano, Loic Kessous, Amaryllis Raouzaïou, Lori Malatesta, Stelios Asteriadis, and Kostas Karpouzis. n.d. *Multimodal emotion recognition from expressive faces, body gestures and speech*. Springer US.
- DaveSGage. 2016. *FFT*. https://commons.wikimedia.org/wiki/File:FFT_of_Cosine_Summation_Function.png.
- de Alencar, V.F.S., and A. Alcaim. 2005. "Transformations of LPC and LSF Parameters to Speech Recognition Features." *Lecture Notes in Computer Science* (Springer, Berlin, Heidelberg) 3686.
- Deng, Li, and Dong Yu. 2014. "Deep Learning: Methods and Applications." *Foundations and Trends in Signal Processing* 7 (3–4): 197–387. <http://research.microsoft.com/pubs/209355/DeepLearning-NowPublishing-Vol7-SIG-039.pdf>.
- Deng, Li, and Xiao Li. 2013. "Machine Learning Paradigms for Speech Recognition: An Overview." *IEEE Transactions on Audio, Speech, and Language*

- Processing* 21 (5): 1060-1089. <https://microsoft.com/en-us/research/publication/machine-learning-paradigms-for-speech-recognition-an-overview>.
- Dhanalakshmi, P., S. Palanivel, and V. Ramalingam. 2011. "Classification of audio signals using AANN and GMM." *Applied Soft Computing* 716-723. doi:10.1016/j.asoc.2009.12.033.
- Dimitrakakis, Christos, and Samy Bengio. 2011. "Phoneme and Sentence-Level Ensembles for Speech Recognition." *Eurasip Journal on Audio, Speech, and Music Processing* 2011 (1): 3. <https://asmp-aurasipjournals.springeropen.com/articles/10.1155/2011/426792>.
- Duarte, Tiago, Rafael Prikladnicki, Fabio Calefato, and Filippo Lanubile. 2014. "Speech Recognition for Voice-Based Machine Translation." *IEEE Software* 31 (1): 26-31. <https://ieeexplore.ieee.org/document/6750466>.
- ETSI. 2000. "Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms." *ETSI ES 201 108 version 1.1.2*. April.
- Fant, Gunnar. 1960. *Acoustic theory of speech production*. Mouton & Co.
- Furui, Sadaoki. 2001. *Digital Speech Processing Synthesis, and Recognition, Second Edition*. CRC Press. doi:10.1201/9781482270648.
- Hannun, Awni, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, et al. 2014. "Deep Speech: Scaling up end-to-end speech recognition." *arXiv: Computation and Language*. <https://arxiv.org/abs/1412.5567>.
- Hansen, John , and Taufiq Hasan. 2015. *Speaker Recognition by Machines and Humans: A tutorial review*. Signal Processing Magazine, IEEE. Vol. 32. IEEE. doi:10.1109/MSP.2015.2462851.
- Hollien, H., W. Majewski, and P. A. Hollien. 1974. "Perceptual identification of voices under normal, stress, and disguised speaking conditions." *The Journal of the Acoustical Society of America*. doi:10.1121/1.1914230.
- Juang, B. H., and Lawrence R. Rabiner. n.d. "Automatic speech recognition—a brief history of the technology development." 6. http://www.ece.ucsb.edu/faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf.
- Juang, B.-H., and L.R. Rabiner. n.d. *Speech Recognition, Automatic: History*. Elsevier.
- Jurafsky, Daniel. 2016. *Speech and Language Processing*.
- King, Simon, Joe Frankel, Karen Livescu, Erik McDermott, Korin Richmond, and Mirjam Wester. 2007. "Speech production knowledge in automatic speech recognition." *Journal of the Acoustical Society of America* 121 (2): 723-742. http://cstr.ed.ac.uk/downloads/publications/2007/king_et_al_review.pdf.
- Kuroiwa, Shingo, Satoru Tsuge, and Fuji Ren. 2009. "Fuzzy Cluster Analysis and its Evaluation Method." *BMFSA Biomedical Fuzzy Systems Association* 1410: 3-10.
- Lippmann, Richard P. 1997. "Speech recognition by machines and humans." *Speech Communication* 22 (1): 1-15. <https://ee.columbia.edu/~dpwe/papers/lipp97-hummach.pdf>.

- n.d. *Meyda, Audio feature extraction for JavaScript*. <https://meyda.js.org/guides/online-web-audio>.
- Microsoft. n.d. *Speaker Recognition API*. Accessed 12 01, 2019. <https://westus.dev.cognitive.microsoft.com/docs/services/563309b6778daf02acc0a508/operations/5645c3271984551c84ec6797>.
- n.d. *Pioneering Speech Recognition*. <http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/speechreco/>.
- Pollack, I., J. M. Pickett, and W. H. Sumby. 1954. *On the identification of speakers by voice*. Journal of the Acoustical Society of America.
- Pruzansky, Sandra. 1963. *Pattern-Matching Procedure for Automatic Talker Recognition*. Vol. 35. The Journal of the Acoustical Society of America. doi:10.1121/1.1918467.
- Pruzansky, Sandra, and Max V Mathews. 1963. *Talker-Recognition Procedure Based on Analysis of Variance*. Vol. 36. The Journal of the Acoustical Society of America. doi:10.1121/1.1919320.
- Rabiner. n.d. "The Acoustics, Speech, and Signal Processing Society. A Historical Perspective." http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/216_historical%20perspective.pdf.
- Ramírez, Javier , and Juan Manuel Górriz. 2011. *Recent Advances in Robust Speech Recognition Technology*. Bentham Books. <https://benthambooks.com/book/9781608051724/>.
- SAWUSCH, JAMES R. 2005. "Acoustic Analysis and Synthesis of Speech." In *The Handbook of Speech Perception*, by David B. Pisoni and Remez Robert E., 7-26. Blackwell Publishing Ltd.
- Schultz, Tanja. n.d. *Multilingual Speech Processing*. Edited by Katrin Kirchhoff. Elsevier.
- Scofield, Jason, and Douglas A. Behrend. 2008. "Learning Words from Reliable and Unreliable Speakers." *Cognitive Development* 23 (2): 278-290. <https://sciencedirect.com/science/article/pii/S088520140800004x>.
- Shrawankar, Urmila, and Vilas M. Thakare. 2013. "Techniques for Feature Extraction In Speech Recognition System : A Comparative Study." *arXiv: Sound*. <https://arxiv.org/abs/1305.1145>.
- Soong, F, A Rosenberg, L Rabiner, and Juang B. 1985. *A vector quantization approach to speaker recognition*. IEEE. doi:10.1109/ICASSP.1985.1168412.
- Tisby, N.Z. 1991. *On the application of mixture AR hidden Markov models to text independent speaker recognition*. Vol. IEEE Transactions on Signal Processing. IEEE. doi:10.1109/78.80876.
- Umeda, Yoshiyuki, Shingo Kuroiwa, Satoru Tsuge, and Fuji Ren. 2004. "Distributed speaker recognition using earth mover's distance." *INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing*. Jeju Island, Korea: ISCA.
- Waibel, Alex. 1989. "Modular Construction of Time-Delay Neural Networks for Speech Recognition." *Neural Computation* 1 (1): 39-46.

- http://isl.anthropomatik.kit.edu/cmu-kit/Modular_Construction_of_Time-Delay_Neural_Networks_for_Speech_Recognition.pdf.
- n.d. *Web Audio API.* https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API.
- Zhang, Yaxin, Jianming Song, and Anton Madievski. 2003. "Tone based speech recognition." *Journal of the Acoustical Society of America* 114 (4): 1715. <http://freepatentsonline.com/6553342.html>.

Image Attribution

Fig. 1. Speaker Verification *Re-mixed "Man Looking at Phone Cartoon Vector.svg* from Wikimedia Commons by Videoplasty.com, CC-BY-SA 4.0" 6

Fig. 2. Physiology of speech *Re-mixed Vocal tract - Attrib- Tavin [CC BY 3.0 (https://creativecommons.org/licenses/by/3.0)]*
This file is licensed under the Creative Commons Attribution 3.0 Unported license.
 6

Fig. 3. The source-filter speech production model. *Re-mixed Vocal tract - Attrib-Tavin [CC BY 3.0 (https://creativecommons.org/licenses/by/3.0)]*
This file is licensed under the Creative Commons Attribution 3.0 Unported license.
 7

Fig. 4. Formants viewed on a spectrogram
File: Spectrogram -iua-.png This file is licensed under the Creative Commons Attribution 2.0 Generic license. https://en.wikipedia.org/wiki/User:ish_ishwar 8

Fig. 5. General speaker recognition system. Adapted from (Campell 1997) (Hansen and Hasan 2015). 9

Fig. 6. Fourier Analysis (bottom) of a Cosine Summation Function (top).
From Wikimedia Commons, the free media repository Author: DaveSGage licensed under the Creative Commons Attribution-Share Alike 4.0 International license https://commons.wikimedia.org/wiki/File:FFT_of_Cosine_Summation_Function.png
 10

6.1 External Sources of Speech Corpus

- 1) *Free Spoken Digit Dataset (FSDD)* - Free Spoken Digit Dataset (FSDD) - 4 speakers, 2,000 recordings (50 of each digit per speaker), English pronunciations. <https://github.com/Jakobovski/free-spoken-digit-dataset>
- 2) *audioMNIST* - 30000 audio samples of spoken digits (0-9) of 60 different speakers. <https://github.com/soerenab/AudioMNIST>