Chapter 2
Speech Processing Background

In this chapter, we review some of the building blocks of speech processing systems. We then discuss the specifics of speaker verification, speaker identification, and speech recognition. We will reuse these constructions when designing privacy-preserving algorithms for these tasks in the reminder of the thesis.

Almost all speech processing techniques follow a two-step process of signal parameterization followed by classification. This is shown in Fig. 2.1.

2.1 Tools and Techniques

2.1.1 Signal Parameterization

Signal parameterization is a key step in any speech processing task. As the audio sample in the original form is not suitable for statistical modeling, we represent it using *features*.

The most commonly used parametrization for speech is Mel-frequency cepstral coefficients (MFCC) (Davis and Mermelstein 1980). In this representation, we segment the speech sample into 25 ms windows, and take the Fourier transform of each window. This is followed by de-correlating the spectrum using a cosine transform, then taking the most significant coefficients.

If x is a frame vector of the speech samples, F is the Fourier transform in matrix form, M is the set of Mel filters represented as a matrix, and D is a DCT matrix, MFCC feature vectors can be computed as

$$MFCC(x) = D\log(M((Fx) \cdot \text{conjugate}(Fx))).$$

M. A. Pathak, *Privacy-Preserving Machine Learning for Speech Processing*,
Springer Theses, DOI: 10.1007/978-1-4614-4639-2_2,

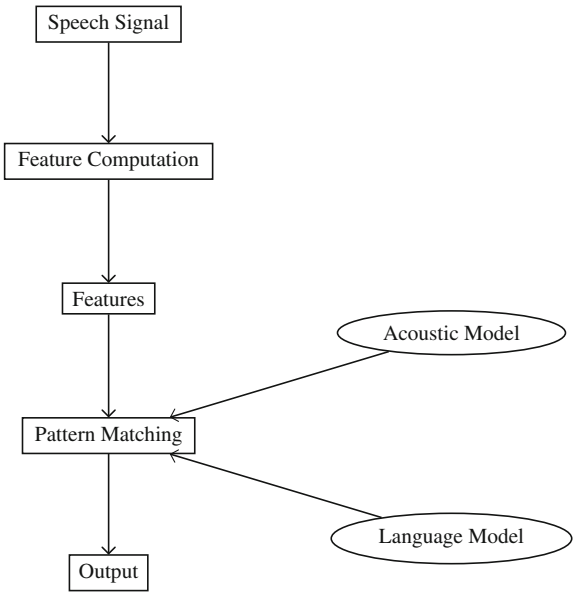

Fig. 2.1 Work flow of a speech processing system

2.1.2 Gaussian Mixture Models

Gaussian mixture models (GMMs) are commonly used generative models for density estimation in speech and language processing. The probability of the model generating an example is given by a mixture of Gaussian distributions (Fig. 2.2).

A GMM λ comprises of M multivariate Gaussians each with a mean and covariance matrix. If the mean vector and covariance matrix of the jth Gaussian are respectively μ_j and Σ_j, for an observation x, we have

$$P(x|\lambda) = \sum_j w_j \mathcal{N}(\mu_j, \Sigma_j),$$

where w_j are the mixture coefficients that sum to one. The above mentioned parameters can be computed using the expectation-maximization (EM) algorithm.

2.1.3 Hidden Markov Models

A hidden Markov model (HMM) (Fig. 2.3), can be thought of as an example of a Markov model in which the state is not directly visible but the output of each state can be observed. The outputs are also referred to as observations. Since observations

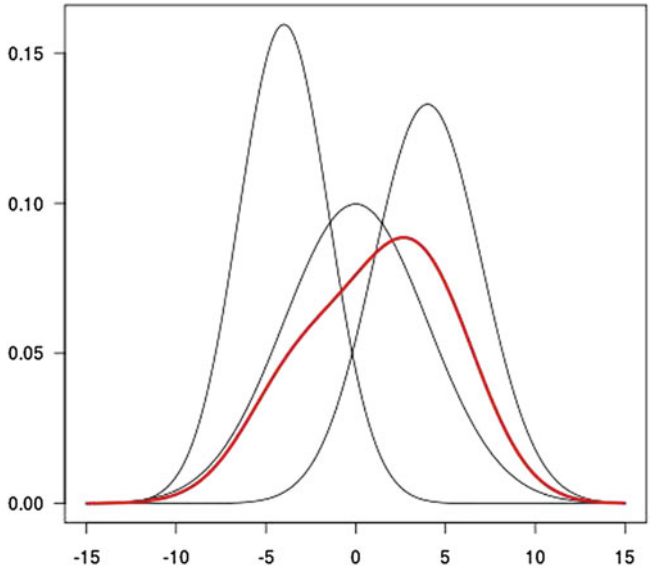

Fig. 2.2 An example of a GMM with three Gaussian components

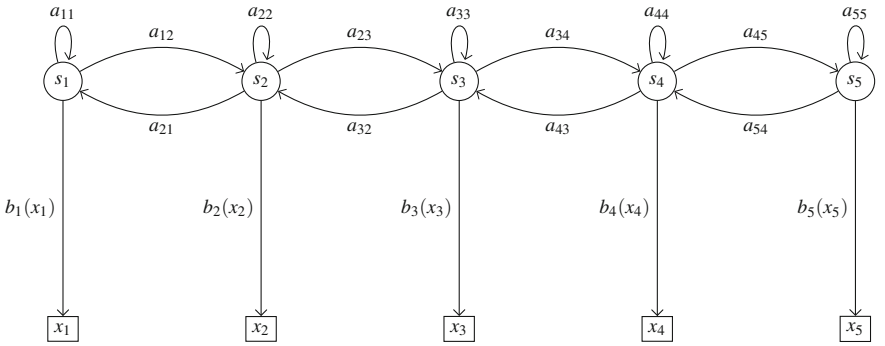

Fig. 2.3 An example of a 5-state HMM

depend on the hidden state, an observation reveals information about the underlying state.

Each HMM is defined as a triple $M = (A, B, \Pi)$, in which

- $A = (a_{ij})$ is the state transition matrix. Thus, $a_{ij} = \Pr\{q_{t+1} = S_j | q_t = S_i\}$, $1 \leq i, j \leq N$, where $\{S_1, S_2, \ldots, S_N\}$ is the set of states and q_t is the state at time t.
- $B = (b_j(v_k))$ is the matrix containing the probabilities of the observations. Thus, $b_j(v_k) = \Pr\{\mathbf{x}_t = v_k | q_t = S_j\}$, $1 \leq j \leq N$, $1 \leq k \leq M$, where $v_k \in V$ which is the set of observation symbols, and $\mathbf{x}_t$ is the observation at time t.

- $\Pi = (\pi_1, \pi_2, \ldots, \pi_N)$ is the initial state probability vector, that is, $\pi_i = \Pr\{q_1 = S_i\}, i = 1, 2, \ldots, N$.

Depending on the set of observation symbols, we can classify HMMs into those with discrete outputs and those with continuous outputs. In speech processing applications, we consider HMMs with continuous outputs where each of the observation probabilities of each state is modeled using a GMM. Such a model is typically used to model the sequential audio data frames representing the utterance of one sound unit, such as a phoneme or a word.

For a given sequence of observations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ and an HMM $\lambda = (A, B, \Pi)$, one problem of interest is to efficiently compute the probability $P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T|\lambda)$. A dynamic programming solution to this problem is the forward algorithm.

2.2 Speaker Identification and Verification

In speaker verification, a system tries to ascertain if a user is who he or she claims to be. Speaker verification systems can be text dependent, where the speaker utters a specific pass phrase and the system verifies it by comparing the utterance with the version recorded initially by the user. Alternatively, speaker verification can be text independent, where the speaker is allowed to say anything and the system only determines if the given voice sample is close to the speaker's voice. Speaker identification is a related problem in which we identify if a speech sample is spoken by any one of the speakers from our pre-defined set of speakers. The techniques employed in the two problems are very similar, enrollment data from each of the speakers are used to build statistical or discriminative models for the speaker which are employed to recognize the class of a new audio recording.

2.2.1 Modeling Speech

We discuss some of the modeling aspects of speaker verification and identification below. Both speaker identification and verification systems are composed of two distinct phases, a training phase and a test phase. The training phase consists of extracting parameters from the speech signal to obtain features and using them to train a statistical model. We typically use MFCC features of the audio data instead of the original samples as they are known to provide better accuracy for speech classification.

Given a speech sample y and a hypothesized speaker s, the speaker verification task can be formulated as determining if y was produced by S. In the speaker identification task, we have a pre-defined set of hypothesized speakers $S = \{S_0, S_1, \ldots, S_K\}$ and we are trying to identify which speaker $s \in S$ would have spoken the speech sample y. Speaker S_0 represents the none of the above case where the speech sample

does not match any of the other K speakers. In this discussion, we consider the single-speaker case where y is assumed to have spoken by only one speaker, please refer to Dunn et al. (2000) for work on detection from multi-speaker speech.

Speaker identification is simply finding the speaker s^* having the highest probability of generating the speech sample.

$$s^* = \underset{s \in S}{\operatorname{argmax}}\, P(y|s).$$

Speaker verification on the other hand can be modeled as the following hypothesis test:

H_0: y is spoken by the speaker s

H_1: y is not spoken by the speaker s

We use a likelihood ratio test (LRT) to decide between the two hypothesis.

$$\frac{P(y|H_0)}{P(y|H_1)} \begin{cases} > \theta & \text{accept } H_0, \\ < \theta & \text{accept } H_1, \end{cases} \tag{2.1}$$

where θ is the decision threshold. The main problem in speaker identification and verification is effectively modeling the probabilities $P(y|s)$ and $P(y|H_0)$.

Modeling for a given speaker s and the hypothesis H_0 is well defined and can be done using training data for that speaker. On the other hand the alternative hypothesis H_1 is open-ended as it represents the entire space of possible speakers except s. There are exactly the same issues with modeling the "none of the above" speaker S_0 in speaker identification. The main approach for modeling the alternative hypothesis is by collecting speech samples from several speakers and training a single model called the universal background model (UBM) (Carey et al. 1991; Reynolds 1997). One benefit of this approach in speaker verification is that a single UBM can be used as the alternate hypothesis for all speakers. There has been work on selection and composition of the right kind of speakers used to train the UBM and also to use multiple background models tailored for a specific set of speakers (Matsui and Furui 1995; Rosenberg and Parthasarathy 1996; Heck and Weintraub 1997).

Selection of the right kind of probability model is a very important step in the implementation of any speech processing task and is closely dependent on the features used and other application details. For text-independent speaker identification and verification, GMMs have been used very successfully in the past. For text-dependent tasks, the additional temporal knowledge can be integrated by using a HMM, but the use of such complex models have not been shown to provide any significant advantage over using GMMs (Bimbot et al. 2004).

For a GMM Φ with M mixture components, the likelihood function density for a D-dimensional feature vector $\mathbf{x}$ is given by

$$P(\mathbf{x}|\Phi) = \sum_{i=1}^{M} w_i P_i(\mathbf{x}),$$

where w_i are the mixture components and $P_i(\mathbf{x})$ is the multivariate Gaussian density parameterized by mean μ_i and covariance Σ_i. Given a collection of vectors $\mathbf{x}$ from the training data, the GMM parameters are estimated using expectation-maximization (EM) algorithm.

For recognition, we assume the feature vectors of a speech sample to be independent. The model probabilities are scaled by the number of feature vectors to normalize for the length of the sample. The log-likelihood of a model Φ for a sequence of feature vectors $X = \{x_1, \ldots, x_T\}$ is given by

$$\log P(X|\Phi) = \frac{1}{T} \log \prod_{t=1}^{T} P(\mathbf{x}_t|\Phi) = \frac{1}{T} \sum_{t=1}^{T} \log P(\mathbf{x}_t|\Phi).$$

Basic speaker verification and identification systems use a GMM classifier model trained over the voice of the speaker. In case of speaker verification, we train a binary GMM classifier using the audio samples of the speaker as one class and a universal background model (UBM) as another class (Campbell 2002). The UBM is trained over the combined speech data of all other users. Due to the sensitive nature of their use in authentication systems, speaker verification classifiers need to be robust to false positives. In case of doubt about the authenticity of a user, the system should choose to reject. In case of speaker identification, we also use the UBM to categorize a speech sample as not being spoken by anyone from the set of speakers.

In practice, we need a lot of data from one speaker to train an accurate speaker classification model and such data is difficult to acquire. Towards this, Reynolds et al. (2000) proposed techniques for maximum *a posteriori* adaptation to derive speaker models from the UBM. These adaptation techniques have been extended by constructing *supervectors* consisting of concatenated means of the mixture components (Kenny and Dumouchel 2004). The supervector formulation has also been used with support vector machine (SVM) classification methods. Other approaches for representing speech samples with noise robustness include factor analysis (Dehak et al. 2011). These methods can be incorporated in our privacy-preserving speaker verification framework.

2.2.2 Model Adaptation

In the above discussion, we considered GMM as a representation of speaker models. In practice, however, we have limited quantity of speech data from individual speakers. It is empirically observed that GMMs obtained from adapting the UBM to speaker data from individual speakers significantly outperform the GMMs trained directly on the speaker data (Bimbot et al. 2004; Reynolds 1997). We present the algorithm for maximum *a posteriori* (MAP) adaptation of the UBM.

The MAP adaptation procedure consists of a sample estimate of the speaker model parameters such as the mixture weights and the means, followed by their

interpolation with the UBM. Given set of enrollment speech data frames $x_1, \ldots, x_T$, we first compute the *a posteriori* probabilities of the individual Gaussians in the UBM $\lambda_U = \{w_i^U, \mu_i^U, \Sigma_i^U\}$. For the ith mixture component of the UBM,

$$P(i|x_t) = \frac{w_i^U \mathcal{N}(x_t; \mu_i^U, \Sigma_i^U)}{\sum_j w_j^U \mathcal{N}(x_t; \mu_j^U, \Sigma_j^U)}. \tag{2.2}$$

Similar to the maximization step of EM, the *a posteriori* probabilities are then used to compute new weights, mean, and second moment parameter estimates.

$$\begin{aligned} w_i' &= \frac{1}{T} \sum_t P(i|x_t), \\ \mu_i' &= \frac{\sum_t P(i|x_t) x_t}{\sum_t P(i|x_t)}, \\ \Sigma_i' &= \frac{\sum_t P(i|x_t) x_t x_t^T}{\sum_t P(i|x_t)}. \end{aligned} \tag{2.3}$$

Finally, the parameters of the adapted model $\lambda_s = \{\hat{w}_i^s, \hat{\mu}_i^s, \hat{\Sigma}_i^s\}$ are given by a convex combination of these new parameter estimates and the UBM parameters as follows.

$$\begin{aligned} \hat{w}_i^s &= \alpha_i w_i' + (1 - \alpha_i) w_i^U, \\ \hat{\mu}_i^s &= \alpha_i \mu_i' + (1 - \alpha_i) \mu_i^U, \\ \hat{\Sigma}_i^s &= \alpha_i \Sigma_i' + (1 - \alpha_i) \left[\Sigma_i^U + \mu_i^U \mu_i^{UT} \right] - \hat{\mu}_i^s \hat{\mu}_i^{sT}. \end{aligned} \tag{2.4}$$

The adaptation coefficients α_i control the amount of contribution of the enrollment data relative to the UBM.

2.2.3 Supervectors with LSH

Campbell et al. (2006) extend the adapted GMM algorithm by constructing a *supervector* (SV) for each speech sample. The supervector is obtained by performing maximum *a posteriori* (MAP) adaptation of the UBM over a single speech sample and concatenating the means of the adapted model. Given the adapted model $\lambda_s = \{\hat{w}_i^s, \hat{\mu}_i^s, \hat{\Sigma}_i^s\}$ with M-mixture components, the supervector sv is given by $(\hat{\mu}_1^s \parallel \hat{\mu}_2^s \parallel \cdots \parallel \hat{\mu}_M^s)$.

This supervector is then used as a feature vector instead of the original frame vectors of the speech sample. The verification is performed using a binary support vector machine (SVM) classifier for each user trained on supervectors obtained from enrollment utterances as instances labeled according to one class and impostor data as instances labeled according to the opposite class. As the classes are usually not

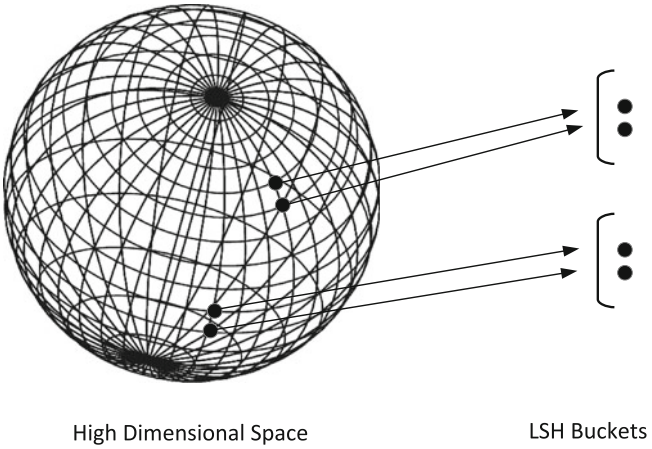

Fig. 2.4 LSH maps similar points to the same bucket

separable in the original space, (Campbell et al. 2006) also use a kernel mapping that is shown to achieve higher accuracy. Instead of using SVMs with kernels, we use k-nearest neighbors trained on supervectors as our classification algorithm. Our motivation for this are, firstly, k-nearest neighbors also perform classification with non-linear decision boundaries and are shown to achieve accuracy comparable to SVMs with kernels (Mariéthoz et al. 2009). Secondly, by using the LSH transformations we discuss below, we reduce the k-nearest neighbors computation to string comparison, which can be easily done with privacy without requiring an interactive protocol.

Locality Sensitive Hashing

Locality sensitive hashing (LSH) (Indyk and Motwani 1998) is a widely used technique for performing efficient approximate nearest-neighbor search. An LSH function $L(\cdot)$ proceeds by applying a random transformation to a data vector x by projecting it to a vector $L(x)$ in a lower dimensional space, which we refer to as the LSH *key* or *bucket*. A set of data points that map to the same key are considered as approximate nearest neighbors (Fig. 2.4).

As a single LSH function does not group the data points into fine-grained clusters, we use a hash key obtained by concatenating the output of k LSH functions. This k-bit LSH function $L(x) = L_1(x) \cdots L_k(x)$ maps a d-dimensional vector into a k-bit string. Additionally, we use m different LSH keys that are computed over the same input to achieve better recall. Two data vectors x and y are said to be neighbors if at least one of their keys, each of length k, matches exactly. One of the main advantages of using LSH is its efficiency: by precomputing the keys, the approximate nearest

neighbor search can be done in time sub-linear to the number of instances in the dataset.

A family of LSH functions is defined for a particular distance metric. A hash function from this family has the property that data points, that are close to each other as defined by the distance metric, are mapped to the same key with high probability. There exist LSH constructions for a variety of distance metrics, including arbitrary kernels (Kulis and Grauman 2009), but we mainly consider LSH for Euclidean distance (E2LSH) (Datar et al. 2004) and cosine distance (Charikar 2002) as the LSH functions for these constructions are simply random vectors. As the LSH functions are data independent, it is possible to distribute them to multiple parties without the loss of privacy.

The LSH construction for Euclidean distance with k random vectors transforms a d-dimensional vector into a vector of k integers, each of which is a number between 0 and 255. The LSH construction for cosine distance with k random vectors similarly transforms the given data vector into a binary string of length k.

2.2.4 Reconstructing Data from LSH Keys

Due to the locality sensitive property, each LSH key provides information about the data point. We consider reconstructing the data point from LSH keys for cosine distance, but our analysis would hold for other distance metrics. A LSH function for cosine and Euclidean distances producing a k bit key is based on k random hyperplanes $\{r_1, \ldots, r_k\}$. Given a data point, we project it using each of the k random hyperplanes and determine the key by the side on which the data point lies.

$$L_i(x) = \begin{cases} 1 & \text{if } r_i^T x \geq 0, \\ 0 & \text{if } r_i^T x < 0. \end{cases}$$

A k bit LSH key provides k bits of entropy. Computing m LSH keys over the same data point reveals mk bits of information.

An LSH key is defined as the sector between two hyperplanes, this is because two vectors x and y lying in this space would have little angular distance $\theta(x, y)$, that corresponds to similarity in terms of cosine distance. By observing multiple keys computed using different LSH functions, we can further localize the data point by the space between any two hyperplanes, which may be from different LSH functions. We show an example of 2-dimensional LSH in Fig. 2.5, application of one LSH function localizes the data point in a wider space, but application of two LSH function further localizes the space between a hyperplane of the first and the second LSH function. As we mentioned in the analysis above, the degree of localization depends on the size of the LSH key, i.e., the number of hyperplanes, and the number of LSH keys. By observing a large number of LSH keys, we can localize a point to lie on a hyperplane.

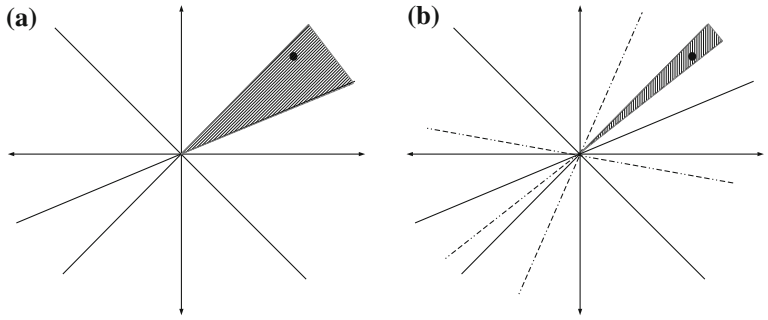

Fig. 2.5 Application of one and two LSH functions. **a** One LSH function (*solid lines*). **b** Two LSH functions (*solid* and *dashed lines*)

As we shall see in later chapters, the information revealed about the data point from the LSH keys causes a problem with respect to privacy. We satisfy the privacy constraints by obfuscating the LSH key. A simple way of doing that is by applying a cryptographic hash function $H[\cdot]$. In this way, we are not able to identify the hyperplanes that localize the data point and use information from multiple LSH keys to reconstruct the data point. We are, however, able to compare if two data points fall in the same localized region, by comparing their hashed keys.

2.3 Speech Recognition

Speech recognition is a type of pattern recognition problem, where the input is a stream of sampled and digitized speech data and the desired output is the sequence of words that were spoken. The pattern matching involves combining acoustic and language models to evaluate features which capture the spectral characteristics of the incoming speech signal. Most modern speech recognition systems use HMMs as the underlying model.

We view a speech signal as a sequence of piecewise stationary signals and an HMM forms a natural representation to output such a sequence of frames. We view the HMM that models the process underlying the observations as going through a number of states, in case of speech, the process can be thought of as the vocal tract of the speaker. We model each state of the HMM using a Gaussian mixture model (GMM) and correspondingly we can calculate the likelihood for an observed frame being generated by that model. The parameters of the HMM are trained over the labeled speech data using the Baum-Welch algorithm. We can use an HMM to model a phoneme which is the smallest segmental unit of sound which can provide meaningful distinction between two utterances. Alternatively, we can also use an HMM to model a complete word by itself or by concatenating the HMMs modeling the phonemes occurring in the word.

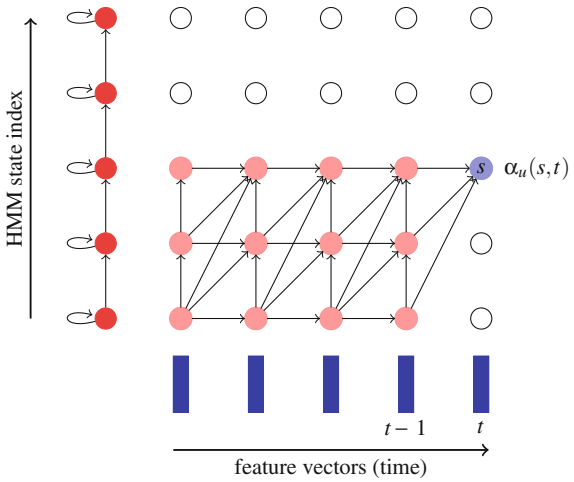

Fig. 2.6 A trellis showing all possible paths of an HMM while recognizing a sequence of frames

A useful way of visualizing speech recognition of an isolated word using an HMM is by a trellis shown in Fig. 2.6. Every edge in the graph represents a valid transition in the HMM over a single time step and every node represents the event of a particular observation frame being generated from a particular state. The probability of an HMM generating a complete sequence of frames can be efficiently computed using the forward algorithm. Similarly, the state sequence having the maximum probability for an observation sequence can be found using the Viterbi algorithm. In order to perform isolated word recognition, we train an HMM for each word in the dictionary as a template. Given a new utterance, we match it to each of the HMMs and choose the one with the highest probability as the recognized word.

References

Bimbot F, Bonastre J-F, Fredouille C, Gravier G, Magrin-Chagnolleau I, Meignier S, Merlin T, Ortega-Garcia J, Petrovska-Delacretaz D, Reynolds D (2004) A tutorial on text-independent speaker verification. EURASIP J Appl Signal Process 4:430–451

Campbell W (2002) Generalized linear discriminant sequence kernels for speaker recognition. In: IEEE international conference on acoustics, speech and signal processing

Campbell WM, Sturim DE, Reynolds DA, Solomonoff A (2006) SVM based speaker verification using a GMM supervector kernel and NAP variability compensation. In: IEEE international conference on acoustics, speech and signal processing

Carey M, Parris E, Bridle J (1991) Speaker verification system using alpha-nets. In: International conference on acoustics, speech and signal processing

Charikar M (2002) Similarity estimation techniques from rounding algorithms. In: ACM symposium on theory of computing

Datar M, Immorlica N, Indyk P, Mirrokni VS (2004) Locality-sensitive hashing scheme based on p-stable distributions. In: ACM symposium on computational geometry, pp 253–262
Davis S, Mermelstein P (1980) Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans Acoust Speech Signal Process ASSP 28(4):357
Dehak N, Kenny P, Dehak R, Dumouchel P, Ouellet P (2011) Front-end factor analysis for speaker verification. IEEE Trans Audio Speech Lang Process 19(4):788–798
Dunn RB, Reynolds DA, Quatieri TF (2000) Approaches to speaker detection and tracking in conversational speech. Digit Signal Process 10:93–112
Heck LP, Weintraub M (1997) Handset-dependent background models for robust text-independen speaker recognition. In: International conference on acoustics, speech and signal processing, vol 2. pp 1071–1074
Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the ACM symposium on theory of computing, pp 604–613
Kenny P, Dumouchel P (2004) Experiments in speaker verification using factor analysis likelihood ratios. In: Odyssey, pp 219–226
Kulis B, Grauman K (2009) Kernelized locality-sensitive hashing for scalable image search. In: IEEE international conference on computer vision, pp 2130–2137
Mariéthoz J, Bengio S, Grandvalet Y (2009) Kernel-Based Text-Independent Speaker Verification, in Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods (eds) J. Keshetand S. Bengio, John Wiley & Sons, Ltd, Chichester, UK. doi: 10.1002/9780470742044.ch12
Matsui T, Furui S (1995) Likelihood normalization for speaker verification using a phoneme- and speaker-independent model. Speech Commun 17(1–2):109–116
Reynolds DA (1997) Comparison of background normalization methods for text-independent speaker verification. In: European conference on speech communication and technology, vol 2. pp 963–966
Reynolds DA, Quatieri TF, Dunn RB (2000) Speaker verification using adapted Gaussian mixture models. Digit Signal Process 10:19–41
Rosenberg AE, Parthasarathy S (1996) Speaker background models for connected digit password speaker verification. In: International conference on acoustics, speech and signal processing

http://www.springer.com/978-1-4614-4638-5

Privacy-Preserving Machine Learning for Speech Processing

Pathak, M.A.

2013, XVIII, 142 p., Hardcover
ISBN: 978-1-4614-4638-5