Fast Search for Common Segments in Speech Signals for Speaker Verification

Michael Gerber, Beat Pfister

Speech Processing Group, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
{gerber,pfister}@tik.ee.ethz.ch

Abstract

We pursue an approach to text-independent speaker verification which takes the decision whether two speech-signals are from the same speaker or not by matching segments that are common to both signals. A previous system using this pattern matching approach has shown good results but the used algorithm to seek for common segments was not satisfactory in terms of speed. In this paper we present an alternative algorithm which we developed to make this search faster. The approach which is based on hidden Markov models is unconstrained with respect to the size of the common segments which can be detected and it is basically language-independent. With the algorithm presented in this paper we could speed up the segment search by a factor of 5 while even improving the segments regarding their matching quality.

Index Terms: hidden Markov model, speaker verification, pattern matching, keyword spotting

1. Introduction

In certain scenarios of speaker verification such as in forensic applications it is necessary to decide from two speech signals S_1 and S_2 whether they have been spoken by the same speaker or not. Pattern matching (PM) is known to be a good approach to take this decision if the two speech signals are equally worded, i.e. in a text-dependent case. In [1] we have presented a way to successfully apply a PM approach to a text-independent scenario. In that approach we first sought common segments, i.e. segments which occur in both speech signals. Such segments had to be at least a few phonemes long in order to be suitable for PM. With that approach we achieved better results than a system which was based on statistical modeling with Gaussian mixture models (GMMs, see e.g. [2]). We think that this is because our PM approach can profit from the sequential order of the frames which is not exploited in a GMM approach apart from the sequential information implicitly contained in the used features (i.e. delta features).

The method we used in [1] to seek common segments was computationally very expensive and therefore was not applicable for longer signals. In this paper we present a new much faster method to seek common segments.

Besides being efficient this algorithm has to meet the following requirements:

- It has to be language-independent and should therefore rely on purely acoustic criteria.
- The algorithm should be able to detect common segments of arbitrary length, i.e. not only predefined words.
- The algorithm should be speaker-independent because generally there is no enrollment data available.

Especially the two first requirements distinguish our method from other approaches to speaker verification which are based on the detection of segments that are common to both speech signals.

For example in [3] a method was presented which uses automatic speech recognition (ASR) transcriptions to seek common, about phoneme-long segments in two speech signals. The common segments are then compared with a PM approach. The method presented in [4] seeks predefined keywords (frequently used words like *and, I, that, yeah, ...*) in the ASR transcriptions of the two signals. Then GMMs are trained for these keywords. A similar approach which is based on ASR as well is described in [5]. Instead of GMMs, word-level HMMs are trained for the keywords. All these approaches depend on ASR which makes them language-dependent. Furthermore, the methods presented in [4] and [5] are restricted to a set of defined keywords.

2. Our speaker verification approach

Our method to decide whether two speech signals are spoken by the same speaker or not includes the following three steps: First common segments are sought in the two speech signals. From these common segments we can extract frame pairs whereby the frames of each pair are phonetically matched. In a second step the probability that the two frames of a pair come from the same speaker is computed for every frame pair. A multilayer perceptron (MLP) is used for this task. In [6], [1] and [7] was shown that an MLP is suitable to calculate this probability and leads to better results than the Euclidean distance which is generally used in PM approaches. From this second step we obtain a series of frame-level probabilities. Finally, a global indicator that the two speech signals were spoken by the same speaker can be calculated from these frame-level probabilities. Further information about our speaker verification approach can be found in [1].

3. New method to seek common segments

In order to find common segments in two signals S_1 and S_2 it would be possible to use a phoneme recognizer to generate sequences of phoneme labels for both signals. Such an approach would be language-dependent since every language has its own set of phonemes. Therefore our algorithm to seek common segments is based on acoustic elements.

The idea is that every acoustic element covers a part of the acoustic space, a subspace. Although not phonetically motivated, the acoustic elements should have the following properties:

- A single subspace should be small enough and it should be positioned in a way that all acoustic vectors located in this subspace are perceived as phonetically very similar.

- The subspaces should be large enough to cover the variability of the same phoneme being produced by different speakers and transmitted over different channels.

We model each such acoustic element with a single HMM state. Thus an acoustic element model does not include temporal aspects. Following the principle in [8], the acoustic element models are trained as described in Section 3.1.

By means of these models it would be possible to determine for the two speech signals S_1 and S_2 the optimal sequences of acoustic elements Q_1 and Q_2. The algorithm to seek common segments could then be done in the symbolic domain. However, the algorithm should be able to detect also segments which are similar but have not been mapped on exactly the same sequence of acoustic elements. It would therefore be necessary to introduce a distance-measure to quantify the similarity of acoustic elements. In order to avoid this we developed an algorithm which performs the search directly in the acoustic domain. This algorithm is described in Section 3.2.

3.1. Training the acoustic elements

The training can roughly be divided into two stages. In the first stage an initial set of acoustic element models $A = \{a_1, a_2 \dots a_R\}$ is generated. This set partitions the acoustic space in an unsupervised manner, i.e. solely the feature distribution determines the subspaces. In the second stage (steps 2 to 4 below) the requirements given in Section 1 are aimed at by forcing all utterances of the same word w_z onto the same sequence Q_z of acoustic elements. The training of the acoustic elements is as follows:

1. We model the feature vectors of all available signals in the training set $\mathcal{S}_{train}$ by means of a fully connected HMM which has as many states as the desired number R of acoustic elements and one mixture per state. This leads to the initial set of acoustic elements.
2. For all signals in $\mathcal{S}_{train}$ we know which words $w_z, z = 1 \dots Z$ were spoken (Z is the number of different words occurring in $\mathcal{S}_{train}$). For every word w_z we collect all signals in which w_z was uttered. We then determine that sequence of acoustic models $Q_z = q_1 q_2 \dots q_m, q_i \in A$ which describes all utterances of this word optimally in the maximum-likelihood sense. We do this in an approximative way by first finding the optimal sequence for each utterance of this word. Out of these sequences we chose the one which describes all other utterances of the same word in the best way.
3. Using the sequences Q_z determined in the previous step, the acoustic element models a_r are re-estimated.
4. If the desired number of mixtures I per acoustic element model is reached, the training is stopped. Otherwise the number of mixtures is doubled and steps 2 to 4 are iterated.

We have empirically determined the optimal number R of acoustic elements to be 128 and the number I of mixtures per element to be 4. It can be seen that the number of elements is higher than the number of phonemes generally used in speech recognizers.

3.2. Common segment detection algorithm

By means of the acoustic element models the common segments in two speech signals S_1 and S_2 are detected as follows: In the first step the optimal sequence B of acoustic elements for S_1 is determined. This is described in 3.2.1. In the second step we seek segments of the signal S_2 which are sufficiently well described by model subsequences of the sequence B. This is described in 3.2.2

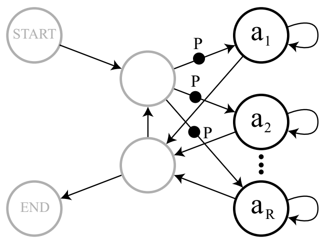

Figure 1: *Markov model to generate the optimal state sequence B of acoustic elements for the signal S_1. Non-emitting states are printed in grey. The black states $a_1 \dots a_R$ correspond to the R acoustic elements as described in 3.1. Some transitions have got additional penalties P.*

3.2.1. Decoding the first signal

To determine the optimal sequence B of S_1 we use an HMM as shown in Figure 1. Each emitting state represents an acoustic element. All emitting states are interconnected via non-emitting states (equivalent to a fully connected HMM). The transitions to the acoustic elements have got a penalty P. This has the effect that the Viterbi decoder favors to remain longer in one state. We have empirically found that an additional log-likelihood of $P = -2$ reduces the length M of B by a factor of about 2 without affecting the modeling-quality of the state sequence B too much. Applying this HMM with the Viterbi algorithm results in a sequence of acoustic elements as e.g.

$$\begin{aligned} B &= a_{66} a_{15} a_{43} \dots a_{97} \\ &= b_1 b_2 b_3 \dots b_M, \quad b_m \in A \end{aligned}$$

For all elements b_m in B we save also the time $t_b(b_m)$ when the element was entered and $t_e(b_m)$ when the element was left.

3.2.2. Find similar segments in the second signal

Now we extend the HMM of Figure 1 with the model sequence B which results in an HMM as shown in Figure 2. This HMM consists of two parts. The part on the right side is identical to the one used in Section 3.2.1. The part on the left side represents the sequence B of acoustic elements found for S_1. These elements constitute first of all a linear HMM. There are additional connections through the non-emitting states that have got again penalties. The motivation for theses penalties P_a and P_b is similar as in keyword-spotting, where penalties are used to tune the operation point of the system to minimize false alarms and missed detections (see e.g. [9]). Here the penalties control the number and the length of the detected segments. In the absence of such penalties the Viterbi decoder would almost never use a transition $b_i \to b_{i+1}$ because S_2 can't be better modeled than with a free sequence of acoustic element models as represented by the right-hand side of the HMM of Figure 2. In this HMM the penalties P_b are higher (i.e. smaller log-likelihood value) than the penalties P_a in order to prevent the detection of very short segments.

The Viterbi decoder delivers a state sequence C through the HMM of Figure 2 for the signal S_2. This sequence C could look like

$$C = a_7 \dots a_{95} b_{18} b_{19} b_{20} b_{21} a_{17} \dots a_{14}$$

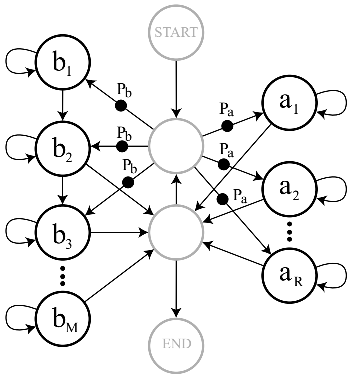

Figure 2: *Special Markov model to find segments of the signal S_2 that match parts of the sequence B. All emitting states (printed in black) are modeled by acoustic elements. Either directly through $a_r \in A$ or indirectly through the b_m. Non-emitting states are printed in grey. P_a and P_b are additional penalties associated with some transitions.*

In sequence C the subsequences consisting of elements b_m represent the desired common segments. In the example above the subsequence of interest is $b_{18}b_{19}b_{20}b_{21}$. With the entry times $t_b(b_{18})$ and the leaving times $t_e(b_{21})$ of B and C we can determine the positions of the segment in the two signals.

A handicap of this method is, that it can detect for a segment of S_2 only one matching segment in S_1 (the other way round is no problem). To alleviate this problem we split the state sequence B into several overlapping windows and perform the algorithm described above for every window of B.

4. Experiments

For our experiments we used recordings with German three-digit numbers. The speech signals were recorded from male speakers over various land-line and mobile telephones. The signals of every speaker were recorded in 15 sessions which were typically spread over one month. The average duration of a spoken three-digit number is 1.5 s.

All experiments shown in this paper were conducted with signals containing 7 three-digit numbers. None of the three-digit numbers of the first signal occurred in the second one. Therefore only common digits can be detected, not whole numbers.

4.1. Data sets

All data was divided into three disjoint speaker subsets. The training set $\mathcal{S}_{train}$ had data from 26 speakers. It was used to train the acoustic element models, the MLPs and the universal background model (UBM) of the baseline system (cf. Section 4.2.2). Therefore, the acoustic element models, the MLPs and the UBM are not speaker-specific. The validation set $\mathcal{S}_{valid}$ contained 10 speakers. It was used to stop the training of the MLP at the right time to prevent overfitting and to optimize the parameters of the algorithm to detect common segments. The test set $\mathcal{S}_{test}$ included 13 speakers.

4.2. Comparison systems

In the tests presented here we used the following systems:

4.2.1. Pattern-Matching with a DTW-like segment search

In previous work we developed a DTW-based (dynamic time warping) algorithm to detect phonetically matched speech segments in S_1 and S_2. Instead of a matrix of local distances we used a probability matrix which shows for every frame-pair (one frame from S_1 and the other from S_2) the probability that the two frames are from the same phoneme. These probabilities were computed by an appropriately trained MLP. Common segments showed therefore as ridges in this probability matrix. We used a DTW-like approach to detect these ridges (see [1]). The computation of the speaker verification score from the found common segments was the same as described in this paper.

4.2.2. Baseline GMM system

As a baseline system we used a GMM system as described in [2]. An optimization on the validation set has shown that the optimal number of Gaussians is 1024. The UBM was trained with data from $\mathcal{S}_{train}$. For each speaker verification trial a speaker model was created by adapting the UBM with the first of the two given signals using a maximum a posteriori adaption (see [10]). More information on the implementation of our GMM system can be found in [1].

4.2.3. System combination

In order to verify if the devised PM system and the GMM system make use of complementary information, we also evaluated a system combination. We used a weighted average of the scores delivered by the two systems. The empirically evaluated optimal weights for the PM system and for the GMM system were found to be $\frac{1}{3}$ and $\frac{2}{3}$, resp. The scores of both systems were normalized to zero mean and unit variance before the fusion.

4.3. Features

We used Mel-frequency cepstral coefficients (MFCCs) from 37.5 ms long frames at a frame rate of 100 Hz. 34 Mel-filters were placed in the frequency range from 100 to 4000 Hz. The cepstral mean of the used signals was subtracted to compensate for linear channel distortions.

For the algorithm to detect common segments (i.e. features for the acoustic elements) and for the computation of the probability matrix mentioned in 4.2.1 we used the MFCCs $0 \dots 12$ plus their first derivatives and for the speaker verification task we used MFCCs $1 \dots 16$ plus their first derivatives.

4.4. Evaluation methods

The following four measures were used to evaluate the algorithms to find common segments:

4.4.1. Matching quality

We generated matched phonetic segmentations (i.e. considering pronunciation variation) of all utterances in $\mathcal{S}_{test}$ with a forced-alignment. Therefore we could check for all frame pairs from the found common segments, whether the two frames were assigned the same phoneme label. If the phoneme labels were matching, we assigned the value 1. If only the phoneme label in a directly neighboring frame of the other signal matched we assigned the value 0.3. All other frame pairs were assigned the value 0. We then calculated the total matching score as the mean of the values assigned to all frame pairs.

4.4.2. *Completeness of the found sequences*

Another interesting criterion is the ratio between the number of detected common segments and the number of common segments which are effectively available in the two speech signals. Therefore we determined potential matches by seeking common string sequences in the graphemic representations of the signals. We required the string sequences to be at least 5 characters long. We used the graphemic representations and not the phoneme labels because the phoneme labels represent pronunciation variants of words whereas the graphemic representations don't. Since it is not possible to predict from a graphemic representation where in the speech signal the corresponding phonemes are exactly located, we used the following approximation: The position was expressed relative to the full length of the graphemic representation. In the speech signal the segment is expected to occur at roughly the same relative position in time. The experiments showed that this approximation is appropriate, since we used only short utterances.

4.4.3. *Speaker Verification Results*

The speaker verification is impaired by a bad matching quality and by a high rate of missed detections of available common segments. If the matching quality is bad, the MLP will be fed with vector pairs of non-matching frames, which will deteriorate its performance. If on the other hand not all common segments are detected, the final decision does not consider all available information.

4.4.4. *Time*

Since the aim was to make the PM approach faster, the execution time is an important criterion as well. We define it as the average execution time for one trial.

4.5. Results

The table below shows the results of our investigations. As far as applicable the results are given for all the performance measures described in Section 4.4. For the *matching quality* and the *completeness* we give in the row *self* the values obtained from signals of the same speaker and in the row *cross* the values obtained from signals of different speakers. The performance of the speaker verification is indicated with the equal error rate (EER).

PM_{HMM} is the system presented in this paper. The system PM_{DTW} is our predecessor system which is described in 4.2.1. *GMM* is the baseline system described in 4.2.2. Finally the results of the system combination as described in 4.2.3 is given.

Table 1: *Results of the various systems.*

System	EER [%]	Matching quality		Complete-ness [%]		Time [s]
		self	cross	self	cross	
PM_{HMM}	3.0	0.72	0.70	82	63	25
PM_{DTW}	3.5	0.69	0.69	88	61	127
GMM	5.0					7
PM_{HMM} + *GMM*	2.2					32

We have seen that the devised PM_{HMM} system to find common segments is about 5 times faster than the predecessor system PM_{DTW}.

For the speaker verification results we conducted significance tests according to McNemar (see for example [11]) and we found that both the PM_{HMM} system alone and and combination of the PM_{HMM} and the *GMM* system performed significantly better than the *GMM* system alone (significance levels < 0.1 %)

5. Conclusions

One problem of a speaker verification system based on PM is the fact that it is more likely to find common segments in signals of the same speaker than in signals of different speakers because there are speaker-specific pronunciations. This results in the much higher *completeness* value of signals of the same speaker. We have seen, however, that the difference of the *completeness* values is smaller for the PM_{HMM} than for the PM_{DTW} system.

With the speed gain achieved we have created the basis to test our approach also on other databases which include much longer speech signals. Furthermore we have seen that the new algorithm with the PM_{HMM} is not only faster but that the matching quality of the found common segments is even a bit better than the one obtained with the PM_{DTW} system.

6. References

[1] M. Gerber, R. Beutler, and B. Pfister, “Quasi text-independent speaker verification based on pattern matching,” in *Proceedings of Interspeech*. ISCA, 2007, pp. 1993–1996.

[2] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, “Speaker verification using adapted Gaussian mixture models,” *Digital Signal Processing*, vol. 10, pp. 19–41, 2000.

[3] A. Corrada-Emmanuel and M. Newman et al., “Progress in speaker recognition at Dragon Systems,” in *Proceedings of the ICSLP, Sydney, Australia*, 1998.

[4] D. E. Sturim and D. A. Reynolds et al., “Speaker verification using text-constrained Gaussian mixture models,” in *Proceedings of the ICASSP*, 2002.

[5] K. Boakye and B. Peskin, “Text-constrained speaker recognition on a text-independent task,” in *Odyssey 2004 - The Speaker and Language Recognition Workshop, Toledo*, June 2004.

[6] U. Niesen and B. Pfister, “Speaker verification by means of ANNs,” in *Proceedings of the ESANN'04, Bruges (Belgium)*, April 2004, pp. 145–150.

[7] M. Gerber, T. Kaufmann, and B. Pfister, “Perceptron-based class verification,” in *Proceedings of Non Linear Speech Processing (NOLISP)*, 2007.

[8] L. R. Bahl and P. F. Brown et al., “A method for the construction of acoustic Markov models for words,” *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 4, pp. 443–452, October 1993.

[9] R. C. Rose and D. B. Paul, “A hidden Markov model based keyword recognition system,” in *Proceedings of the ICASSP*, vol. 1, 1990, pp. 129–132.

[10] J.-L. Gauvain and C.-H. Lee, “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains,” *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–299, April 1994.

[11] L. Gillick and S. Cox, “Some statistical issues in the comparison of speech recognition algorithms,” in *Proceedings of the ICASSP*, 1989, pp. 532–535.