Open access • Book Chapter • DOI:10.1007/3-540-44887-X_75

A speaker pruning algorithm for real-time speaker identification — Source link

Tomi Kinnunen, Evgeny Karpov, Pasi Fränti

Institutions: University of Eastern Finland

Published on: 09 Jun 2003 - Lecture Notes in Computer Science (Springer, Berlin, Heidelberg)

Topics: Speaker diarisation, Speaker recognition, Pruning (decision trees) and TIMIT

A Speaker Pruning Algorithm for Real-Time Speaker Identification

Tomi Kinnunen, Evgeny Karpov, Pasi Fränti

University of Joensuu, Department of Computer Science
P.O. Box 111, 80101 Joensuu, Finland
{tkinnu, ekarpov, franti}@cs.joensuu.fi

Abstract. Speaker identification is a computationally expensive task. In this work, we propose an iterative speaker pruning algorithm for speeding up the identification in the context of real-time systems. The proposed algorithm reduces computational load by dropping out unlikely speakers as more data arrives into the processing buffer. The process is repeated until there is just one speaker left in the candidate set. Care must be taken in designing the pruning heuristics, so that the correct speaker will not be pruned. Two variants of the pruning algorithm are presented, and simulations with TIMIT corpus show that an error rate of 10 % can be achieved in 10 seconds for 630 speakers.

1 Introduction

The *speaker identification* task is defined as follows: given an unknown speaker and a set of N candidate speakers, find the most similar speaker among the candidates [1, 2]. More precisely, this is a *closed-set* speaker identification task which means that the unknown speaker is assumed to be one of the candidate speakers.

In general, a speaker identification system usually consists of the following four parts: feature extraction, speaker modeling, pattern comparison, and decision logic. Given an unknown speakers voice sample and the stored candidate speaker models, the system first computes feature vectors from the given speech sample. Then the feature vectors are compared against all of the N models using the pattern comparison algorithm. The result of this phase is a list of *match scores*, which can be either similarity or dissimilarity values. The decision logic finally makes a *one-out-of-N* decision, e.g. selects the speaker with maximum degree of similarity.

Speech user interfaces and speaker adaptation methods in speech recognition systems are examples of a potential application of speaker identification technology. In such systems, the identification time must be minimized so that the system works in real-time, or near real-time. Speaker identification is a computationally expensive problem [1, 2]. The identification time is dominated by two factors: the number of speakers (N) and the number of unknown speakers feature vectors (M). Identification requires $N \cdot M$ distance (or similarity) computations. By reducing the number of speakers or feature vectors, identification time can be significantly reduced. *Silence detection* is a simple example of reducing the number of feature vectors [8].

In this work we aim to reduce the number of computations by reducing the number of candidate speakers. The basic idea is that when a given amount of new data (feature vectors) comes in, we drop a certain amount of candidate speakers away. The process is repeated until finally just one speaker is left in the candidate set. We assign this speaker to be the most similar to the unknown speech sample.

2 Principle of Speaker Pruning

The principle of the proposed speaker pruning is illustrated in Fig. 1. The ellipses represent the models of speakers in the speaker database, and the "**x**" dots are the feature vectors of the unknown speaker. Initially, all the speakers are in the database. When more data comes in, a few of the most dissimilar speakers are pruned away, and they are not anymore used in the pattern comparisons. The process is repeated until there is only one speaker left in the candidate set. The decision of the speaker identity is the last speaker left in the set.

The speaker pruning framework described above is in a general form. The following issues must be taken in the consideration in the actual implementation:

1. what are the features,
2. what is the presentation of speaker models, and what is the pattern comparison method,
3. what is the pruning criterion (algorithm),
4. how many new vectors are read from input buffer prior to next pruning,
5. how many speakers are pruned at each iteration.

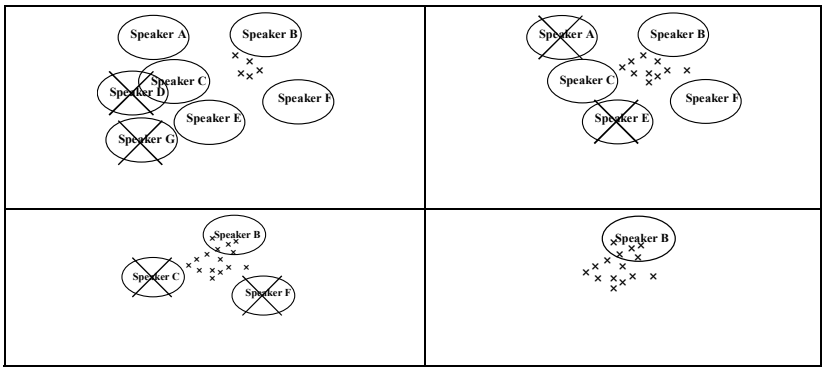

Fig. 1. Illustration of speaker pruning.

In this work the feature vectors are composed of the 12 lowest mel-frequency cepstral coefficients (MFCC) computed using 27 mel-spaced filters. The 0^{th} cepstral coefficient is excluded. Analysis frame is windowed by a 30 ms Hamming window, and frame shift is 10 ms. The signal is pre-emphasized by the filter $H(z) = 1 - 0.97z^{-1}$. Before the feature extraction, silent frames are removed based on simple short-term

energy thresholding [3]. All speech sample durations in this paper refer to silence-removed speech.

All speakers are modeled by a codebook [4, 7] of 64 vectors using the generalized Lloyd algorithm (GLA) as the clustering method [5]. The pattern comparison method is the *average quantization error* (or *distortion*) $D(X, C)$ between the test vector sequence X and the codebook C [7]. The speaker with the minimum quantization error is selected as the best matching candidate.

The design issues 3, 4 and 5 are discussed in the next Section in detail.

3 Speaker Pruning Algorithm

A pruning algorithm with two variants is proposed. The variants are referred to as *Static* and *Adaptive pruning*. These will be described in Sections 3.1 and 3.2. The details of the control parameters of the algorithms will be discussed in Section 3.3. The following notations will be used:

X	Feature vectors of the unknown speaker.
C_i	The model (codebook) of ith speaker.
$D(X, Y)$	Dissimilarity of vector sequences X and Y.
M	The number of new vectors read at each iteration.
K	The number of pruned speakers at each iteration.

3.1 Static pruning

The basic idea in *Static pruning* is to maintain an ordered list of the best matching speakers. At each iteration, K worst matching speakers are pruned out from the list. As new vectors arrive from the input buffer, the dissimilarity values between the augmented vector set and the remaining speaker models are updated. Note that, in practice, the re-evaluation of the dissimilarities can be done fast by using cumulative counts of distances. The pseudocode of the method is given in Fig. 2.

3.2 Adaptive Pruning

In the second variant, *Adaptive pruning*, the pruning criterion is *data-driven*: the number of speakers to be pruned depends on the current distribution of the dissimilarity values between the unknown speaker and the remaining speaker models. Based on the mean value μ and standard deviation σ of the dissimilarity distribution, a *pruning threshold* θ is set, and all speakers above this threshold are pruned out. After pruning, the dissimilarity distribution changes, and its mean value and standard deviation must be updated to obtain the updated pruning threshold. The pseudocode is given in Fig. 3.

Let $C = \{C_1,\ldots,C_N\}$ be the set of all speaker models ;
Let $X = \emptyset$;
WHILE ($C \neq \emptyset$ **AND** vectors left in input buffer) **DO**
  Insert M new vectors from input buffer to set X ;
  Re-evaluate dissimilarities $D(X, C_i)$ for all C_i in C ;
  Remove K most dissimilar models from C ;
END
RETURN arg $\min_i$ { $D(X, C_i) \mid C_i \in C$ } ;

Fig. 2. Static pruning algorithm.

Let $C = \{C_1,\ldots,C_N\}$ be the set of all speaker models ;
Let $X = \emptyset$;
WHILE ($C \neq \emptyset$ **AND** vectors left in input buffer) **DO**
  Insert M new vectors from input buffer to set X ;
  Re-evaluate dissimilarities $D(X, C_i)$ for all C_i in C ;
  Compute μ and σ of the distribution { $D(X, C_i) \mid C_i \in C$ } ;
  Let $\theta = \mu + \eta\,\sigma$ be the pruning threshold ;
  Remove all speakers i from C satisfying $D(X, C_i) > \theta$;
END
RETURN arg $\min_i$ { $D(X, C_i) \mid C_i \in C$ } ;

Fig. 3. Adaptive pruning algorithm.

3.3 Controlling the Pruning

Both the static and adaptive variants have a parameter M, which will be referred to as *pruning interval*. It simply specifies the amount of vectors read from the input buffer before the next pruning.

The variants differ in the way the number of pruned speakers is defined. The static variant has a parameter K which specifies the number of speakers pruned at each iteration. In the adaptive variant, the number of pruned speakers depends on the distribution of the current dissimilarity values. The parameter η determines the degree of the thresholding: the larger η is, the less speakers are pruned, and vice versa.

The formula for calculating the pruning threshold has the following interpretation. From visual inspections of speaker dissimilarity values on TIMIT corpus we found out that the speaker dissimilarity distribution follows more or less a Gaussian curve, as shown in Fig. 4. Because of this, the pruning threshold corresponds to a certain *confidence interval* of the normal distribution, and η specifies its width. Speakers above the upper value of confidence interval will be pruned. For instance, if $\eta = 1$ then speakers above the 68 % confidence interval will be pruned; that is approximately 16 % of the speakers.

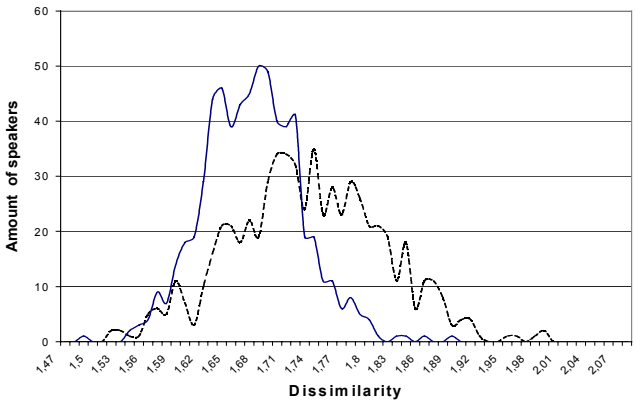

Fig. 4. Examples of typical dissimilarity value distributions from TIMIT corpus (N=630 speakers).

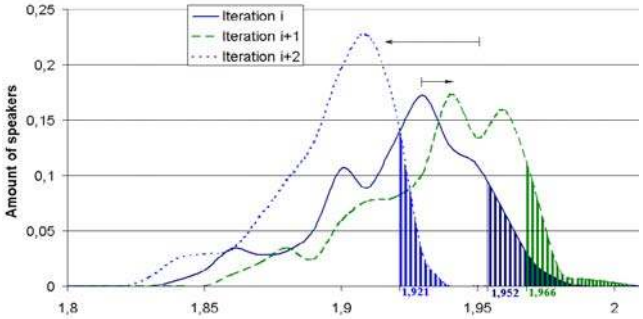

Fig. 5. Moving of the dissimilarity values and the pruning threshold with time.

An example how the distributions and thresholds change over time is shown in Fig. 5. Our general observation is that the variance of the dissimilarity values decreases with time, which can be explained by the fact that the "outlier" speakers are pruned out and the remaining speakers are close to the unknown speaker.

4 Experiments

For experiments, we used the American English TIMIT corpus [6] with all of the 630 speakers included. The length of training data (silence-removed speech) for speaker models was on average 8.8 s. We used 8 kHz sampling frequency and 16-bit resolution. The features were extracted as described in Section 2.

4.1 Results

We consider the trade-off between identification error rate and average time spent on the identification. By lenghtening the pruning interval or by decreasing the number of pruned speakers, we expect smaller error rate, but with the cost of increased identification time. From several runs with different parameter combinations we can plot the error rate as a function of identification time. These curves are shown in Figures 6 and 7 for the static and adaptive variants, respectively.

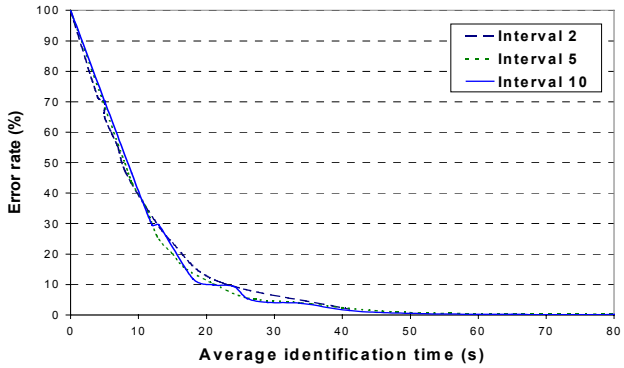

Fig. 6. Evaluation of the static variant using pruning intervals M = 2, 5, 10.

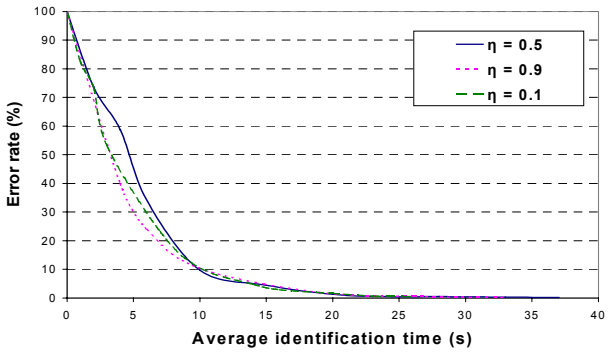

Fig. 7. Evaluation of the adaptive variant for parameter values η = 0.1, 0.5, 0.9.

Next, we compared the two variants by selecting the best curves from each variant. These are shown in Fig. 8. The pruning interval for the static variant is $M = 5$ vectors (or 50 milliseconds), and the parameter η for the adaptive variant is $\eta = 0.9$.

An experiment without any pruning and using all available test data was also carried out. In this case, for the $N = 630$ speakers, only one speaker was misclassified and the error rate is therefore about 0.15 %, with average identification time about 230 seconds. The high identification rate is due to fact that TIMIT is recorded in a noise-free laboratory environment.

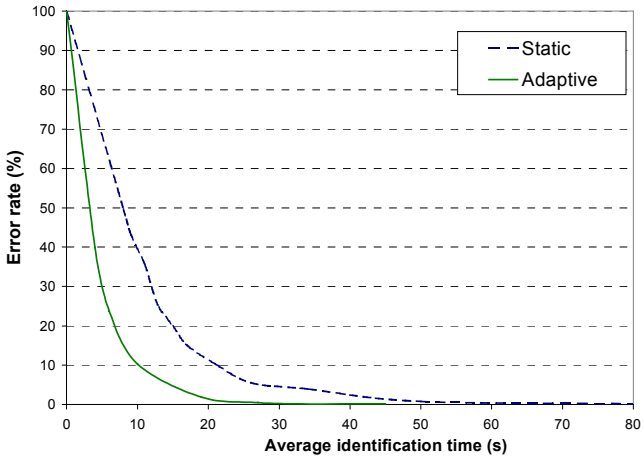

Fig. 8. Comparison of the proposed variants.

4.2 Discussion

The following observations are made from the results. In the static variant, the pruning interval has only little effect on the error rate or identification time. With an average identification time of 10 seconds, the error rate is 40 % in the best case. With the identification time of 50 seconds, the error rate drops below 0.5 %.

In the adaptive variant, the parameter η has some effect on the performance for small identification times. For high identification times the curves do not show significant differences. An error rate of 10 % is reached in 10 seconds identification time. With 25 seconds, an error rate less than 0.5 % is achieved. This is half of the time of the static variant for the same error rate.

We also ran a few tests with larger pruning intervals and η-parameters (up to M = 30 and η = 2.0) but the results were poor. For instance, using η = 2.0, an error rate of 0.3 % took more than 100 seconds to reach. We decided to include only the best results here.

The results for best parameter combinations for both variants are shown in Fig. 8. It is evident that the adaptive variant works better in general. It reaches lower error rate with the same identification time. The adaptive method reaches an error rate of 0.46 % with 24 seconds of speech, whereas the static method spends over 60 seconds to reach the same error rate. Compared to the full search which reaches the error rate 0.15 % in 230 seconds, the speed-up is significant in both cases.

In the case of only 3 seconds spent for identification, the error rate for the adaptive method is 53 %. Therefore, the methods need further optimization in order to work in real-time with acceptable error rate.

5 Conclusions

In this paper we have presented a method for reducing the computational load in real-time speaker identification systems. Two variants of the algorithm with different pruning heuristics were presented, and their performance on the TIMIT corpus was studied. The adaptive method outperforms the static variant in every case. With the adaptive method, a 10 % error rate can be reached in 10 seconds with 630 speakers.

Both variants are easy to implement. In the future, we plan to extend the algorithm to use time-dependent values for M, K and η parameters. For instance, pruning interval M should be initially large so that the unknown speakers feature vector distribution stabilizes, and then it should be supressed gradually to make the identification faster.

References

1. J. Campbell, "Speaker Recognition: A Tutorial," *Proc. IEEE*, **85** (9), pp. 1437-1462, 1997.
2. S. Furui: "Recent Advances in Speaker Recognition," *Pattern Recognition Letters*, **18**, pp. 859-872, 1997.
3. J.R. Deller Jr., J.H.L. Hansen, and J.G. Proakis, *Discrete-time Processing of Speech Signals*, Macmillan Publishing Company, New York, 2000.
4. T. Kinnunen, and I. Kärkkäinen, "Class-Discriminative Weighted Distortion Measure for VQ-Based Speaker Identification," *Proc. Joint IAPR International Workshop on Statistical Pattern Recognition (S+SPR2002)*, pp. 681-688, Windsor, Canada, 2002.
5. Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Comm.*, **28**(1), pp. 84-95, 1980.
6. *Linguistic Data Consortium*, http://www.ldc.upenn.edu/
7. F.K. Soong, A.E. Rosenberg, B.-H. Juang, and L.R. Rabiner, "A Vector Quantization Approach to Speaker Recognition," *AT&T Technical Journal*, **66**, pp. 14-26, 1987.
8. D. Burileanu, L. Pascalin, C. Burileanu, M. Puchiu, "An Adaptive and Fast Speech Detection Algorithm", *Proc. 3rd International Workshop on Text, Speech and Dialogue (TSD 2000)*, pp. 177-182, Brno, Czech Republic, 2000.