Real-Time Speaker Identification*

Tomi Kinnunen, Evgeny Karpov, and Pasi Fränti

University of Joensuu, Joensuu, Finland
Department of Computer Science
`{tkinnu,ekarpov,franti}@cs.joensuu.fi`

Abstract

In speaker identification, most of the computation originates from distance or likelihood computations between the feature vectors of the unknown speaker and the models in the database. The identification time depends on the number of feature vectors, their dimensionality, the complexity of the speaker models and the number of speakers. In this paper, we focus on optimizing vector quantization (VQ) based speaker identification. We reduce the number of test vectors by pre-quantizing the test sequence prior to matching, and the number of speakers by pruning out unlikely speakers during the identification process. The best variants are then generalized to Gaussian mixture model (GMM) based modeling also. We obtain a speed-up factor of 16:1 with VQ-based system, and 34:1 with GMM-based system with a minor degradation in the identification error rate.

1. Introduction

Speaker *identification* is a task of finding the best-matching speaker for unknown speaker from a database of known speakers [3]. Speaker *verification*, in turn, consists of deciding whether a given unknown speech sample was uttered by a claimed identity. In this paper we focus on reducing the computational load of the identification task.

A large number of methods have been proposed for speeding up the *verification* task, in particular the Gaussian mixture model (GMM) [14] based systems. The *UBM-adaptation approach* for training GMM-based systems [13] induces a hierarchy between the UBM and speaker-depended GMMs which enables efficient GMM scoring. Other optimizations of GMM include *clustering of GMM components* [2], *Gaussian shortlisting* [1], *input vector reordering* combined with *beam-search pruning* [12], *decimation of the input vectors* [5] and *tree-structured GMMs* [10,17].

GMM-based speaker modeling is widely used and has many advantages. However, *vector quantization* (VQ) based modeling [15] is computationally less demanding and simpler to implement than GMM, and according to our experiments, it gives equal or even better performance than GMM. Based on these arguments, VQ-based recognition is well suited for practitioners.

Most of the computation time in speaker identification is spent on the match score computations. *Speaker pruning* [6,12,18] can be used to reduce the search space by dropping out unlikely speakers "on the fly" as more speech data arrives. In this work, we compare and optimize several speaker pruning variants, and propose also a novel variant called *confidence-based speaker pruning*.

We also reduce the number of test sequence vectors by silence removal and pre-quantization, and demonstrate how pre-quantization and pruning can be combined. A *vantage-point tree* (VPT) [16] is used for indexing the code vectors for speeding up the nearest neighbor search. We also generalize the methods from VQ to GMM.

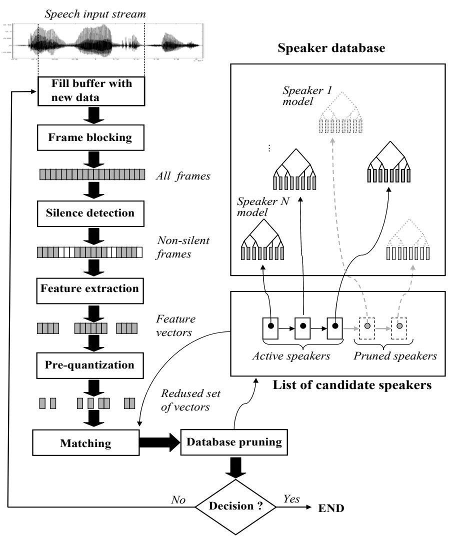

Figure 1: Diagram of the real-time identification system.

2. VQ-Based Speaker Identification

Any speaker identification system includes a *feature extractor* which converts the raw signal into a sequence of feature vectors $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$. Commonly used features include *mel-cepstrum* (MFCC) and *LPC-cepstrum* (LPCC) [4], which both measure the short-term spectral envelope.

In the training phase, a *speaker model* is created by clustering the training feature vectors into disjoint groups by any clustering algorithm. The *LBG algorithm* [8] is widely used due to its efficiency and simple implementation. However, other clustering methods can also be considered; a comparative study can be found in [7]. The result of clustering is a set $C = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M\}$ of vectors, called a *codebook*. The codebook of each speaker is stored in the system database.

In the identification phase, unknown speaker's vectors X are matched against the codebooks $\{C_1, \ldots, C_N\}$ and the codebook giving the best match is selected. Usually, the match score $D(X, C_i)$ is computed by mapping each test vector $\boldsymbol{x} \in X$ to its nearest neighbor in C_i, and accumulating or averaging these quantization distortions [15]. The speaker yielding the smallest distortion is selected as the winner.

Simple nearest neighbor search from C_i requires $O(T \cdot M)$ distance calculations. The computation of $D(X, C_i)$ is repeated for

*Extended version of the work has been submitted to *IEEE Transactions on Speech and Audio Processing*.

$i = 1, \ldots, N$, giving total number of $O(N \cdot T \cdot M)$ distance calculations. Notice that the feature extraction increases the time with an additive constant, since it needs to be done only once. Therefore we have discarded it here.

In order to speed up the nearest neighbor search from a codebook, we utilize *vantage-point tree* (VPT) [16] for indexing the code vectors. VPT is a balanced binary tree for searching in metric spaces. It cannot be used to index the codebooks themselves, since $D(\cdot, \cdot)$ does not satisfy the triangular inequality as required by the VPT.

3. Real-Time Speaker Identification

The proposed system architecture is depicted in Fig. 1. The input stream is processed in short buffers, which are divided into frames. The frames are passed through a silence detector that uses a simple energy-based thresholding. Feature extraction is performed for the remaining frames, and the vectors are then pre-quantized to a smaller representative sequence. Pre-quantized vectors are then matched against *active speakers* only. After the match scores for each speaker are obtained, a number of speakers are pruned out so that they are not included anymore in the matching on the next iteration. The process is repeated until there is no more input data, or there is only one speaker left in the list of active speakers.

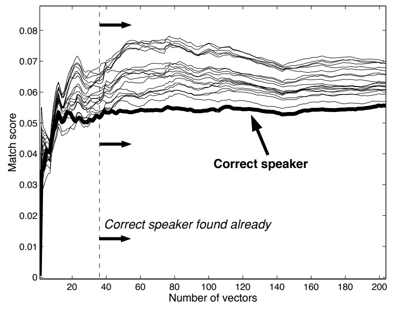

Figure 2: Match score saturation (20 speakers from TIMIT).

By *pre-quantization* (PQ) we mean reducing the bit rate of the input sequence without compromizing the accuracy of the representation. In practise, adjacent feature vectors are correlated, and they can be harmful for recognition [5].

We consider four PQ variants: (1) *random subsampling*, (2) *averaging*, (3) *decimation*, and (4) *clustering-based PQ*. In random subsampling and averaging, the input buffer is processed in non-overlapping segments of $M > 1$ vectors. In random subsampling, each segment is represented by a randomly selected vector. In averaging, the representative vector is the centroid (average vector) of the segment. In decimation, we simply take every Kth vector of the test sequence, which corresponds to performing feature extraction with a smaller frame rate. In clustering-based PQ, we first partition the sequence X into K clusters by the LBG clustering algorithm [8], and input the reduced vector set to the matching function.

3.1. Speaker Pruning

A number of speakers are pruned out at each iteration. One must fix the number of new vectors (non-silent, pre-quantized) used for updating the match scores. We call this the *pruning interval*. Also, the *pruning criterion* must be decided, for which we consider several variants.

Figure 2 shows an example how the average distortion $D(X, C_i)$ develops with time. The bold line represents the correct speaker. In the beginning, the scores oscillate; however, when more vectors are obtained, the scores stabilize around the expected values of the distortions. An important observation is that a small amount of feature vectors is enough to rule out most of the speakers from the set of candidates.

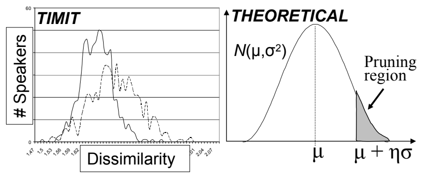

Figure 3: Left: Match score distributions from TIMIT. Right: Illustration of the pruning threshold.

Algorithm 1 Confidence-Based Pruning (CP)

```
$A := \{1, 2, \ldots, N\}$ ; $X := \emptyset$ ;
for $i := 1, \ldots, N$ do
    $D_{prev}[i] := 0$ ; stable[$i$] := false ;
end for
while ($|A| > 1$) and (vectors in input buffer) do
    Insert $M$ new vectors into buffer $X$ ;
    Update $D(X, C_i)$ for all $i \in A$ ;
    Update pruning threshold $\Theta$ ;
    for $i \in A$ do
        $D_{curr}[i] := D(X, C_i)$ ;
    end for
    for $i \in A$ do
        if ( $|1 - D_{prev}[i]/D_{curr}[i]| < \epsilon$ ) then stable[$i$] = true ;
        if (stable[$i$]) and ($D_{curr}(X, C_i) > \Theta$) then $A = A \setminus \{i\}$ ;
        $D_{prev}[i] := D_{curr}[i]$ ;
    end for
end while
Decision: $i^* = \arg\min_i \{D(X, C_i) | i \in A\}$ ;
```

We consider three pruning variants: *static*, *hierarchical*, and *adaptive pruning*. The idea in static pruning (SP) [6] is to maintain an ordered list of the best matching speakers. At each iteration, K worst matching speakers are pruned out. More data is obtained, and the match scores for the remaining speakers are updated.

In hierarchical pruning (HP) [18], for each speaker two codebooks are stored: a *coarse* and a *detail* codebook. Both codebooks are generated from the same training data, but the size of the coarse model is much smaller than the size of the detail model. Test vectors are first scored against the coarse models, and a number of speaker are pruned out. Then, the match scores of the remaining speakers are refined using the detail models.

In adaptive pruning (AP) [6], a *pruning threshold* Θ is computed from the distribution of the match scores and all speakers satisfying $D(X, C_i) > \Theta$ are pruned out. The threshold is defined as $\Theta = \mu_D + \eta \cdot \sigma_D$, where μ_D and σ_D are mean and standard deviation of the active speakers' match scores, and η is a control parameter. The larger η is, the less speakers are pruned, and vice versa (see Fig. 3).

3.2. Confidence-Based Pruning (CP)

We propose a novel pruning variant called *confidence-based pruning*. In CP, only speakers whose match scores have stabilized are considered for pruning. If the match score is poor but it oscillates, the speaker can still change its rank and become the winner. Thus, we remove only speakers that have both stabilized and whose match score (average distortion) exceeds the pruning threshold. The pseudocode of the method is given in Algorithm 1. The set A contains the indices of the active speakers, M is the pruning interval, ϵ is the stabilization threshold, and Θ is the pruning threshold.

Algorithm 2 PQ + Static Pruning Combined (PQP)

```
$A := \{1, 2, \ldots, N\}$ ;
Read new data into buffer $X$ ;
$\hat{X}$ := LBG-Clustering($X, M$) ;
Compute $D(\hat{X}, C_i)$ for all $i \in A$ ;
Prune out $K$ worst speakers from $A$ based on $\{D(\hat{X}, C_i)\}$;
Compute $D(X, C_i)$ for all $i \in A$ ;
Decision: $i^* = \arg\min_i \{D(X, C_i) | i \in A\}$ ;
```

3.3. Combining PQ and Pruning (PQP)

Clustering-based PQ and static pruning can be combined (see Algorithm 2). First, the whole input data is pre-quantized using the LBG algorithm [8]. Using the match scores for the quantized data, K worst scoring speakers are pruned out, and the decision is made by comparing the unquantized data with the remaining models.

4. Experiments

For the experiments, the *TIMIT* corpus [9] was used for parameter tuning and the results were then validated using the *NIST 1999 speaker recognition evaluation corpus* [11]. TIMIT has been recorded in a sound-proof room with a high-quality microphone, whereas NIST is recorded over telephone network. For consistency, TIMIT was downsampled to 8 kHz (NIST has 8 kHz sampling rate).

For TIMIT, we used all 630 speakers, and for NIST, we selected the male subset containing 230 speakers. For NIST, the 1-speaker test segments from the same telephone line with mixed handsets were used for identification. The standard MFCC's [4] were used as the acoustic front-end. The experiments were performed on a cluster of two computers having 2.8 GHz CPUs and 1024 MB of RAM each.

4.1. Baseline System (TIMIT)

First, a few preliminary tests were carried out on TIMIT for tuning the silence threshold and an appropriate number of MFCCs. With the optimized silence threshold, about 12 % of the frames were classified as silent, and the identification time improved by about 10 % without degradation in accuracy. The error rates stabilized around 10-15 coefficients. For the rest of the experiments, the number of MFCCs was fixed to 12.

Table 1 summarizes the performance of the baseline system. The last row shows the results for using the training vectors directly as the speaker model. The accuracy improves with increasing model size; however, the results detoriate due to the overfitting effect when all data is used as the model. For the rest of the experiments, codebook size was fixed to 64.

Table 1: Performance of the baseline system (TIMIT).

Codebook size	Error rate (%)	Avg. identific. time (s)
8	10.5	0.33
64	0.48	2.07
512	0.32	12.9
No model	1.59	23.7

4.2. Pre-Quantization and Pruning (TIMIT)

Parameters of the PQ variants were optimized for each variant separately, and the best time-error trade-off curves are summarized in Fig. 4. The clustering-based PQ gives the best results while the averaging gives the worst results. In general, PQ can be used to reduce the running time about to 50 % of the full search with minor degradation in accuracy.

Next, we evaluated the performance of the speaker pruning variants with the pre-quantization turned off. The parameters of the variants were optimized separately, and the best results for each variant are compared in Fig. 5. The adaptive variant gives the best performance, whereas the static variant gives the worst performance.

The best PQ (clustering-based) and pruning (adaptive) variants, as well as the PQP are compared in Fig. 6. We observe that in time-critical applications the PQ gives slightly better results than pruning. The combination (PQP) gives the best result as expected.

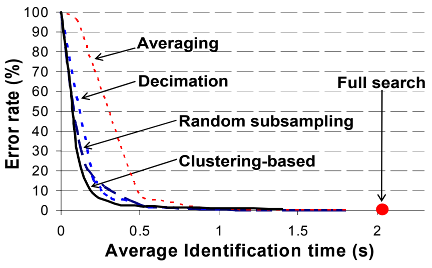

Figure 4: Comparison of the PQ methods (TIMIT).

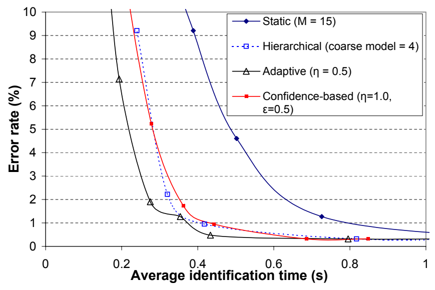

Figure 5: Comparison of the pruning variants (TIMIT).

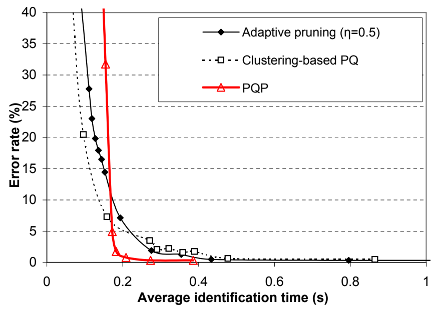

Figure 6: Comparison of PQ, pruning, and PQP (TIMIT).

4.3. Validation with NIST and Generalization to GMM

We generalize the best pre-quantization and pruning variants to GMM modeling as follows: instead of the likelihood $p(X|\text{GMM}_i)$, we use $-p(X|\text{GMM}_i)$ so that the match score becomes a dissimilarity. For GMM, diagonal-covariance models were trained using the basic EM algorithm [4].

For PQ, we selected the clustering variant, for pruning the AP variant, and for the combination the PQP. The best results for both VQ and GMM are summarized in Table 2. We can make the following observations:

- VQ is computationally more efficient than GMM. This is explained by the higher number of parameters in the GMM.
- The error rates are approximately of the same order for both VQ and GMM (around 17-19 %).

Table 2: Summary of the best results on the NIST 1999 corpus.

Setup	Vector quantization (VQ)				Gaussian mixture model (GMM)			
	Model size	Error rate (%)	Time (s)	Speed-up factor	Model size	Error rate (%)	Time (s)	Speed-up factor
Baseline	64	18.06	2.92	1:1	64	17.34	9.58	1:1
PQ		18.20	0.62	5:1		18.79	0.73	13:1
Pruning		19.22	0.48	6:1		19.36	0.82	12:1
PQP		18.06	0.50	6:1		17.34	0.94	10:1
Baseline	128	17.78	5.80	1:1	128	17.05	18.90	1:1
PQ		18.93	0.64	9:1		18.20	0.84	23:1
Pruning		18.49	0.86	7:1		17.34	2.88	7:1
PQP		17.78	0.67	9:1		17.63	1.34	14:1
Baseline	256	17.34	11.40	1:1	256	16.90	37.93	1:1
PQ		18.20	0.70	16:1		18.50	1.11	34:1
Pruning		17.49	1.46	8:1		17.48	5.78	7:1
PQP		17.49	0.90	13:1		18.06	2.34	16:1

- The speed-up factor of PQ increases with the model size as expected. Relative speed-up is higher for GMM than for VQ. Improvement of pruning depends less on the model size.
- The best speed-up factor with VQ is 16:1 increasing the error rate from 17.34 % to 18.20 %. For GMM, the corresponding speed-up factor is 34:1 with the increase of error from 16.90 % to 18.50 %.

Clearly the algorithms work well for both corpora, even though the absolute error rates are higher for NIST. The algorithms formulated for VQ-modeling generalize directly to GMM-modeling. In fact, the speed-up factors are even better for GMM than for VQ. The optimized systems are close to each other both in time and accuracy, and we cannot state that one of the models would be better than the other in terms of time/error trade-off. However, the ease of implementation, numerical stability and lower memory usage make VQ more attractive for practical systems. In fact, prototype implementation for Symbian series60 operating system for mobile devices is currently in progress.

5. Conclusions

A real-time speaker identification system based has been proposed. We used TIMIT corpus for tuning the algorithm parameters, and validated the results using the NIST-1999 corpus. The best methods were also generalized to GMM-based identification, which showed a speed up of 34:1 with minor degradation in the accuracy.

It must be noted that the identification rates on the NIST corpus are much worse than on the TIMIT corpus, which is explained by the differences in the recording conditions. We have not used any channel compensation methods nor delta coefficients, which also explains the relatively high error rate of NIST. In future, we plan to extend our experiments with a better acoustic front-end.

6. Acknowledgements

The work of E. Karpov was supported by the National Technology Agency of Finland (TEKES).

7. References

[1] R. Auckenthaler and J.S. Mason. Gaussian selection applied to text-independent speaker verification. In *Proc. Speaker Odyssey 2001*, pages 83–88, Crete, Greece, 2001.

[2] H.S.M. Beigi, S.H. Maes, J.S. Sorensen, and U.V.Chaudhari. A hierarchical approach to large-scale speaker recognition. In *Proc. Eurospeech 1999*, pages 2203–2206, Budapest, Hungary, 1999.

[3] J. Campbell. Speaker recognition: a tutorial. *Proc. of the IEEE*, 85(9):1437–1462, 1997.

[4] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing: a Guide to Theory, Algorithm, and System Development.* Prentice-Hall, New Jersey, 2001.

[5] J.McLaughlin, D.A. Reynolds, and T. Gleason. A study of computation speed-ups of the GMM-UBM speaker recognition system. In *Proc. Eurospeech 1999*, pages 1215–1218, Budapest, Hungary, 1999.

[6] T. Kinnunen, E. Karpov, and P. Fränti. A speaker pruning algorithm for real-time speaker identification. In *Proc. AVBPA 2003*, pages 639–646, Guildford, UK, 2003.

[7] T. Kinnunen, T. Kilpeläinen, and P. Fränti. Comparison of clustering algorithms in speaker identification. In *Proc. IASTED Int. Conf. Signal Processing and Communications (SPC 2000)*, pages 222–227, Marbella, Spain, 2000.

[8] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Comm.*, 28(1):84–95, 1980.

[9] Linguistic data consortium, April 2004. `http://www.ldc.upenn.edu/`.

[10] M. Liu, E. Chang, and B. q. Dai. Hierarchical gaussian mixture model for speaker verification. In *Proc. ICSLP 2002*, pages 1353–1356, Denver, Colorado, USA, 2002.

[11] A. Martin and M. Przybocki. The NIST 1999 speaker recognition evaluation - an overview. *Digit. Sign. Proc.*, 10(1-18):1–18, 2000.

[12] B.L. Pellom and J.H.L. Hansen. An efficient scoring algorithm for gaussian mixture model based speaker identification. *IEEE Sign. Proc. Lett.*, 5(11):281–284, 1998.

[13] D.A. Reynolds, T.F. Quatieri, and R.B. Dunn. Speaker verification using adapted gaussian mixture models. *Digit. Sign. Proc.*, 10(1):19–41, 2000.

[14] D.A. Reynolds and R.C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Trans. on Speech and Audio Processing*, 3:72–83, 1995.

[15] F.K. Soong, A.E. Rosenberg A.E., B.-H. Juang, and L.R. Rabiner. A vector quantization approach to speaker recognition. *AT & T Tech. J.*, 66:14–26, 1987.

[16] J. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Inform. Proc. Lett.*, 40:175–230, 1991.

[17] B. Xiang and T. Berger. Efficient text-independent speaker verification with structural gaussian mixture models and neural network. *IEEE Trans. on Speech and Audio Processing*, 11:447–456, September 2003.

[18] Z.Pan, K. Kotani, and T. Ohmi. An on-line hierarchical method of speaker identification for large population. In *NORSIG 2000*, Kolmården, Sweden, 2000.