Pattern Recognition Letters 32 (2011) 1604–1617

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Comparison of clustering methods: A case study of text-independent speaker modeling

Tomi Kinnunen, Ilja Sidoroff, Marko Tuononen, Pasi Fränti *

Speech and Image Processing Unit, School of Computing, University of Eastern Finland, P.O. Box 111, FI-80101 Joensuu, Finland

ARTICLE INFO

Article history:
Received 10 September 2010
Available online 2 July 2011
Communicated by G. Borgefors

Keywords:
Clustering methods
Speaker recognition
Vector quantization
Gaussian mixture model
Universal background model

ABSTRACT

Clustering is needed in various applications such as biometric person authentication, speech coding and recognition, image compression and information retrieval. Hundreds of clustering methods have been proposed for the task in various fields but, surprisingly, there are few extensive studies actually comparing them. An important question is how much the choice of a clustering method matters for the final pattern recognition application. Our goal is to provide a thorough experimental comparison of clustering methods for text-independent speaker verification. We consider parametric Gaussian mixture model (GMM) and non-parametric vector quantization (VQ) model using the best known clustering algorithms including iterative (*K*-means, random swap, expectation–maximization), hierarchical (pairwise nearest neighbor, split, split-and-merge), evolutionary (genetic algorithm), neural (self-organizing map) and fuzzy (fuzzy C-means) approaches. We study recognition accuracy, processing time, clustering validity, and correlation of clustering quality and recognition accuracy. Experiments from these complementary observations indicate clustering is not a critical task in speaker recognition and the choice of the algorithm should be based on computational complexity and simplicity of the implementation. This is mainly because of three reasons: the data is not clustered, large models are used and only the best algorithms are considered. For low-order models, choice of the algorithm, however, can have a significant effect.

1. Introduction

Text-independent speaker recognition (Bimbot et al., 2004; Campbell, 1997; Kinnunen and Li, 2010) aims at recognizing persons from their voice. It consists of two different tasks: *identification* and *verification*. The identification task aims at finding the best match (or a set of potential matches) for an unknown voice from a speaker database. The goal of verification task, in turn, is either to accept or reject a claimed identity given by speaking (“I am **Tomi**, verify me”), or by typing a personal identification number (PIN), for instance.

Speaker recognition process is illustrated in Fig. 1. When a new person is enrolled into the system, the audio signal is first converted into a set of feature vectors. Although short-term spectral features (Huang et al., 2001) are sensitive to noise and channel effects, they provide better recognition accuracy than prosodic and “high-level” features (Reynolds et al., 2004), and are therefore used in this study. Following feature extraction, a *speaker model* is trained and added into the database. In the matching phase, feature vectors are extracted from the unknown sample and compared with the model(s) in the database, providing a similarity score. To increase robustness to signal variability, recent solutions use sophisticated speaker model compensation (Burget et al., 2007; Kenny et al., 2008) and score normalization using background speakers (Auckenthaler et al., 2000; Reynolds et al., 2000). Finally, the normalized score is compared with a threshold (verification), or the best scoring speaker(s) is selected as such (identification).

A number of different classifiers have been studied for speaker recognition; see Ramachandran et al. (2002), Kinnunen and Li (2010) for an overview. Speaker models can be divided into *generative* and *discriminative* models. Generative models characterize the distribution of the feature vectors *within* the classes (speakers), whereas discriminative modeling focuses on modeling the decision boundary *between* the classes. For generative modeling, *vector quantization* (VQ) (Burton, 1987; He et al., 1999; Hautamäki et al., 2008; Kinnunen et al., 2006; Soong et al., 1987; Tran and Wagner, 2002) and *Gaussian mixture model* (GMM) (Reynolds and Rose, 1995; Reynolds et al., 2000) are commonly used. For discriminative training, artificial neural networks (ANNs) (Farrell et al.,

Abbreviations: ANN, artificial neural network; DET, detection error trade-off; EER, equal error rate; EM, expectation maximization; FAR, false acceptance rate; FRR, false rejection rate; FCM, fuzzy C-means; GMM, Gaussian mixture model; GA, genetic algorithm; MAP, maximum *a posteriori*; MFCC, mel-frequency cepstral coefficient; PNN, pairwise nearest neighbor; RS, random swap; SOM, self-organizing map; SM, split-and-merge; SVM, support vector machine; UBM, universal background model; VQ, vector quantization.

* Corresponding author.
E-mail addresses: tkinnu@cs.joensuu.fi (T. Kinnunen), isido@cs.joensuu.fi (I. Sidoroff), mtuonon@cs.joensuu.fi (M. Tuononen), franti@cs.joensuu.fi (P. Fränti).

0167-8655/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2011.06.023

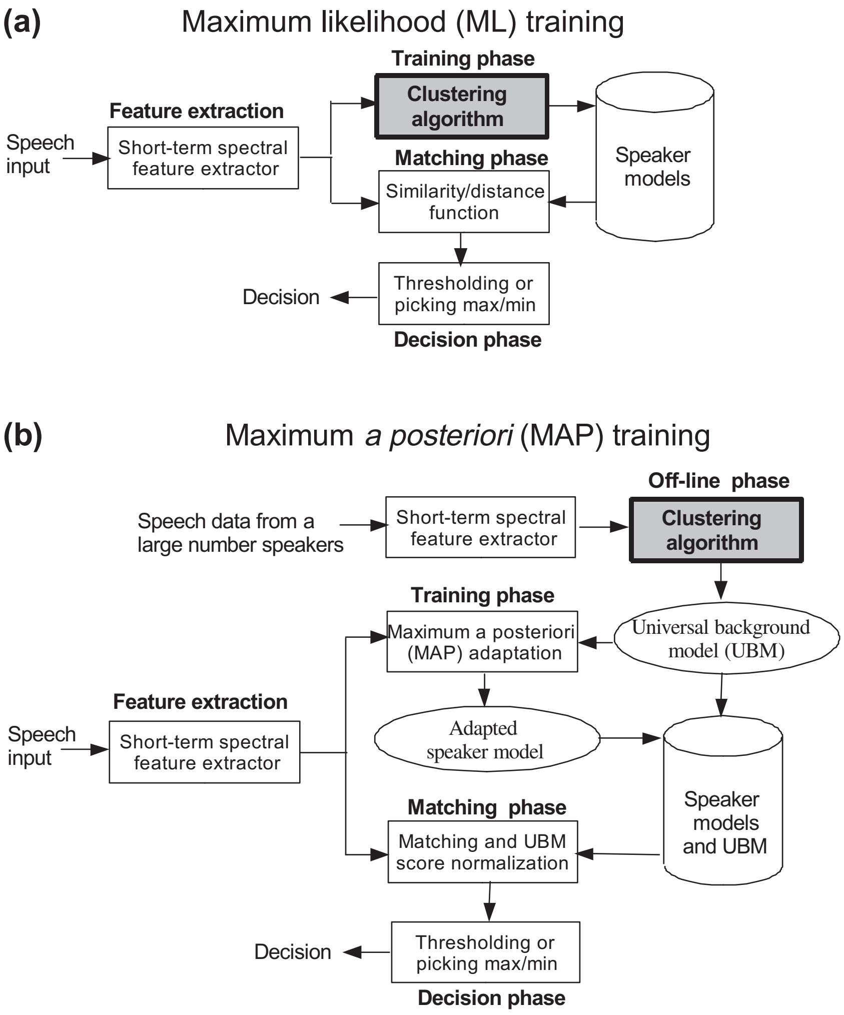
(a)
Maximum likelihood (ML) training
Training phase
Clustering algorithm
Feature extraction
Speech input
Short-term spectral feature extractor
Matching phase
Similarity/distance function
Speaker models
Decision
Thresholding or picking max/min
Decision phase
(b)
Maximum a posteriori (MAP) training
Off-line phase
Speech data from a large number speakers
Short-term spectral feature extractor
Clustering algorithm
Training phase
Maximum a posteriori (MAP) adaptation
Universal background model (UBM)
Feature extraction
Speech input
Short-term spectral feature extractor
Adapted speaker model
Speaker models and UBM
Matching phase
Matching and UBM score normalization
Decision
Thresholding or picking max/min
Decision phase

Fig. 1. System diagram of a spectral feature speaker recognizer, the focus of this study. Clustering methods are characterized here by their clustering quality, resulting speaker recognition accuracy, time consumption of the modeling, and usability aspects. There are two common ways to train speaker models, (a) *maximum likelihood* (ML) that trains the model using feature vectors of the speaker, and (b) *maximum a posteriori* (MAP) that uses in addition a *universal background model* (UBM) to generate a robust model.

1994; Yegnanarayana and Kishore, 2002) and, more recently, support vector machines (SVMs) (Campbell et al., 2006a,b) are representative techniques.

In the past few years, research community has also focused on combining generative and discriminative models, leading to *hybrid* models. In particular, GMMs are extensively used for mapping variable-length vector sequences into fixed-dimensional supervectors (Campbell et al., 2006b; Dehak et al., 2011; Lee et al., 2008) that are used as features in SVM. Parallel to, and in conjunction with this research trend, significant recent advances have also been made on intersession variability compensation of the supervectors (Burget et al., 2007; Dehak et al., 2011; Kenny et al., 2008). A representative class of such techniques is *factor analysis* (FA) model in the GMM supervector space (Dehak et al., 2011; Kenny et al., 2008). Both the hybrid GMM–SVM approaches and the factor analysis models have excellent performance especially under severe channel and session mismatches. However, due to the additional training steps required for constructing the GMM front-end and, subsequently, the session variability models, the supervector methods typically require at least an order of magnitude more development data compared to traditional generative models (Hasan and Hansen, in press; Hautamäki et al., 2008; Reynolds et al., 2000), and are therefore much more CPU-intensive. Careful selection of the various modeling data sets is also a critical step.[1]

1.1. Relevance of clustering in a large-scale pattern recognition problem

In this paper, we focus on the training methodology of two classical generative speaker models, GMM and VQ, for two reasons. Firstly, these methods underlie both the traditional maximum likelihood (or minimum distortion) trained speaker models (Burton, 1987; He et al., 1999; Kinnunen et al., 2006; Soong et al., 1987; Tran and Wagner, 2002), their maximum *a posteriori* adapted extensions using *universal background model* (UBM) priors (Hasan and Hansen, in press; Hautamäki et al., 2008; Reynolds et al., 2000) and, importantly, also the recent hybrid GMM–SVM and

[1] In practice, the various datasets need to be selected according to the expected conditions of the actual application data, reflecting the channel conditions, noise levels, as well as the speaker population (e.g. native language of speakers). Additional care must be taken when the same speakers (but possibly different utterances) are reused in background modeling and score normalization. The degrees of variability used in the NIST speaker recognition evaluation datasets (http://nist.gov/itl/iad/mig/sre.cfm) increases every year and proper selection of training datasets is critical.

FA models (Campbell et al., 2006b; Dehak et al., 2011; Kenny et al., 2008; Lee et al., 2008). The way the underlying generative model is trained will have a major effect to the performance of all these methods. Secondly, while there are good guidelines for composing a balanced and representative training set for background modeling – see the recent study (Hasan and Hansen, in press) and references therein – the question of how to model the generative model itself has received only little attention in literature. Typically, Gaussian mixture models, which are pertinent not only in speaker recognition but in all speech applications and general audio classification tasks (Bagci and Erzin, 2007), are trained using the expectation–maximization (EM) algorithm, or, in the case of vector quantization, the *K*-means algorithm.

Better clustering algorithms have been introduced after *K*-means and EM (Jain, 2010), in terms of preventing local minima, being less sensitive to parameter setup and providing faster processing. Even though several literature surveys exist (Jain et al., 1999; Jain, 2010; Milligan, 1981; Theodoridis and Koutroumbas, 2009), only a few extensive comparisons are available in image processing (Fränti and Virmajoki, 2006) and text retrieval (Steinbach et al., 2000) but none in speaker recognition. In clustering research, new methods are usually compared in terms of clustering quality. But should better clustering quality improve the recognition accuracy of the full pattern recognition system? Overall, given the long history of clustering research (Jain, 2010), existence of thousands of clustering methods, we feel that it is time to review the choice of clustering methodology in a large-scale, real-world pattern recognition problem involving tens of dimensions and hundreds of pattern classes of highly noisy data. In our view, text-independent speaker recognition is a representative application. The main goal of this paper is to bridge some of the gap between theoretical clustering research and large-scale pattern recognition applications, by focusing to an important practical design question: choice of clustering methodology. Before representing the research hypotheses, we first review the role of GMM and VQ clustering methods in our target application.

1.2. Review of clustering methods in speaker recognition

The VQ model (*centroid model*) is a collection of prototype vectors determined by minimizing a *distance*-based objective function. GMM is a *model*-based approach (Meilă and Heckerman, 2001) where the data is assumed to follow Gaussian mixture distribution parameterized by mean vectors, covariance matrices and mixing weights. For a fixed number of clusters, GMM has more free parameters than VQ. Their main difference is the cluster overlap in GMM. In fact, VQ can be seen as a special case of the GMM in which the posterior probabilities have been hardened, and unit variance is assumed in all clusters. Similarly, *K*-means algorithm (Linde et al., 1980) can be considered as a special case of the *expectation maximization* (EM) algorithm for GMM (Bishop, 2006).

The VQ model was first introduced to speaker recognition in (Burton, 1987; Soong et al., 1987) and the GMM model in (Reynolds and Rose, 1995). GMM remains a core component in state-of-the-art speaker recognition whereas VQ is usually seen as a simplified variant of GMM. GMM combined with UBM (Reynolds et al., 2000) is the *de facto* reference method (Fig. 1b). The role of VQ, on the other hand, has been mostly in reducing the number of training or testing vectors to reduce the computational overhead. VQ has also been used as a pre-processor for ANN and SVM classifiers in (Louradour and Daoudi, 2005; Um et al., 2000) to reduce the training time and for speeding up the GMM-based verification in (Kinnunen et al., 2006; Roch, 2006). In (Lei et al., 2005) VQ is used for partitioning the feature space into local decision regions modeled by SVMs to increase accuracy. Despite its secondary role, VQ gives comparable accuracy to GMM (Brew and Cunningham, 2010; Kinnunen et al., 2009) when equipped with a MAP adaptation (Hautamäki et al., 2008). The computational benefits over GMM are important in small-footprint implementations such as mobile devices (Saastamoinen et al., 2005). Recently, similar to hybrids of GMM and SVM (Campbell et al., 2006b), combination of VQ with SVM has also been studied (Brew and Cunningham, 2010).

Fuzzy clustering (Dunn, 1974) is a compromise between VQ and GMM models. It retains the simplicity of VQ while allowing soft cluster assignments using a membership function. Fuzzy extensions of both VQ (Tran and Wagner, 2002) and GMM (Tran et al., 1998) have been studied in speaker recognition. For a useful review, refer to (Tran, 2000). Another recent extension of GMM is based on nonlinear warping of the GMM density function (Wu et al., 2009). These methods, however, lack formulation for the background model adaptation (Reynolds et al., 2000), which is an essential part of modern speaker verification relying on MAP training (Fig. 1b).

The *model order* – number of centroid vectors in VQ or Gaussian components in GMM – is an important control parameter in both VQ and GMM. Typically the number varies from 64 to 2048, depending on the chosen features and their dimensionality, number of training vectors, and the selected clustering model (VQ or GMM). In general, increasing the number of clusters improves recognition accuracy, but it levels off after a certain point due to overfitting. From the two clustering models, VQ was found to be less sensitive to the choice of the number of clusters in (Stapert and Mason, 2001) when trained without the UBM adaptation. The model order in both VQ and GMM needs to be carefully optimized for the given data to achieve good performance (Kinnunen et al., 2009).

The choice of the clustering method, on the other hand, has been much less studied. Usually *K*-means (Linde et al., 1980) and expectation–maximization (EM) (Bishop, 2006; McLachlan and Peel, 2001) methods have been used, although several better clustering methods exist (Fränti and Virmajoki, 2006). This raises the questions of which clustering algorithm should be chosen, and whether the choice between VQ or GMM model matters. Regarding the choice between these models, experimental evidence is diverse. GMM has been shown to perform better for small model orders (Stapert and Mason, 2001), but the difference vanishes when using larger model order (He et al., 1999; Kinnunen et al., 2006; Stapert and Mason, 2001). However, GMM has been reported to work better than VQ only when cluster-dependent covariance matrices were used but perform worse when a shared covariance matrix was used (Reynolds and Rose, 1995). Several authors have used GMM derived from the VQ model for faster training (Kolano and Regel-Brietzmann, 1999; Pelecanos et al., 2000; Singh et al., 2003). All these observations are based on the maximum likelihood (ML) training of speaker models though.

Two recent studies include more detailed comparisons of GMM and VQ (Kinnunen et al., 2009; Hanilci and Ertas, 2011). In (Kinnunen et al., 2009) the MAP trained VQ outperformed MAP-trained GMM for longer training data (2.5 min) but the situation was reversed for 10-second speech samples. The study of Hanilci and Ertas (2011) focused on the choice of dissimilarity measure (city-block, euclidean, Chebychev) in VQ and two different clustering initializations (binary LBG splitting (Linde et al., 1980) versus random selection). Differences in the identification and verification tasks, as well as ML versus MAP training were also considered. The authors found the distance measure and the number of clusters to be more important than the choice of the *K*-means initialization. ML-trained models performed better with the shorter NTIMIT data in speaker identification, whereas MAP-trained models (both GMM and VQ) worked better on longer training segments (NIST 2001). Regarding the choice between GMM and VQ, they performed equally well on the NIST 2001 verification task, regardless whether trained by ML or MAP. However, in the identification task, MAP-trained GMM outperformed MAP-trained VQ, on both corpuses.

A recent study (Brew and Cunningham, 2010) compares MAP-trained GMM and VQ models when used as front-end features for SVM. From the two corpuses, GMM variant outperformed VQ on the YOHO corpus with short utterances, whereas VQ performed slightly better on the KING corpus with longer free-vocabulary utterances.

1.3. Research objectives and hypotheses

Existing literature lacks extensive comparison between different clustering algorithms that would be useful for practitioners. The existing comparisons in speaker recognition study only a few methods, use different features and datasets preventing meaningful cross-comparisons. Even in (Hanilci and Ertas, 2011; Kinnunen et al., 2009), only the basic EM and *K*-means algorithms were studied. Thus, extensive comparison of better clustering algorithms is still missing.

In the experimental section of this paper, we consider the GMM and VQ models both in the maximum likelihood (ML) and maximum a posteriori (MAP) training setting, without additional SVM back-end, inter-session compensation or score normalization (Auckenthaler et al., 2000). Focusing on this computationally feasible core component enables detailed study of generative model training methodology without re-training the full recognition system from scratch every time the background models are changed; the same rationale was chosen recently in (Hasan and Hansen, in press).

In the experiments, we consider both controlled laboratory quality speech (TIMIT corpus) and noisy conversional telephony speech (NIST 1999 and NIST 2006 corpuses). Our main evaluation criteria are the recognition accuracy, processing time and ease of implementation. We aim at answering the following questions:

1. Is clustering needed or would random sub-sampling be sufficient?
2. What is the best algorithm in terms of quality, efficiency and simplicity?
3. What is the difference between the accuracy of the VQ and GMM models?

It was hypothesized in (Kinnunen et al., 2000) that a clustering would be required but the choice of clustering algorithm would not be critical. A possible explanation is that the speech data may not have a clustering tendency (Kinnunen et al., 2001). These observations were based on a small 25-speaker laboratory-quality data collected using the same microphone and read sentences. In this paper, we aim at confirming these hypotheses via extensive large scale experiments. Since the main advantage of speaker recognition over other biometric modalities is possibility for low-cost *remote authentication*, we experiment using realistic telephony data including different handsets, transmission lines, GSM coding and environmental noises. The two NIST corpuses used in the study include 290,521 (NIST 1999) and 53,966 (NIST 2006) verification trials including 539 and 816 speakers, respectively. Furthermore, in NIST 2006 corpus, all the verification trials are from highly mismatched channel conditions. This makes it a very challenging pattern recognition problem.

Regarding the difference between the VQ and GMM models, our results reveal insights which are not obvious, and sometimes contradict previous understanding based on literature. For example, even though the models are of similar quality in terms of average speaker verification accuracy (*equal error rate*), their performance differs systematically at the extreme cases where small false acceptance or false rejection errors are required.

2. Clustering models as speaker models

2.1. Problem formulation

We consider a training set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i = (x_i^{(1)}, \ldots, x_i^{(d)}) \in \boldsymbol{R}^d$ are the *d*-dimensional feature vectors. In the *centroid-based* model, also known as the *vector quantization* (VQ) model, the clustering structure is represented by a set of *code vectors* known as the *codebook*, which is denoted here as $C = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K\}$, where $K \ll N$. The size of the codebook (K) is considered as a control parameter. For a fixed *K*, the clustering problem can be defined as an optimization problem, in which the goal is to find a codebook *C* that minimizes a given objective function. Here we use the *mean square error* (MSE):

$$\mathrm{MSE}(X, C) = \frac{1}{N}\sum_{i=1}^{N} \min_{1 \leqslant k \leqslant K} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2, \tag{1}$$

where $\|\boldsymbol{x}\|^2 = \sum_{j=1}^{d} x_i^2$ denotes the squared Euclidean norm.

In *Gaussian mixture model* (GMM), each cluster is represented by three parameters: mean vector $\boldsymbol{\mu}_{k,}$ covariance matrix Σ_k, and the mixing weight w_k. By considering *K* Gaussian components, the clustering objective function can be defined as the *average log-likelihood*:

$$L(X, \Theta) = \frac{1}{N}\sum_{i=1}^{N} \log \sum_{k=1}^{K} w_k N(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{2}$$

where $\Theta = \{\mu_k, \Sigma_k, w_k\}_{k=1}^{K}$ denotes the model parameters, and $N(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \Sigma_k)$ is the multivariate Gaussian density function with parameters $\boldsymbol{\mu}_k$ and Σ_k. The mixing weights w_k are constrained to be positive and to sum up to 1.

In early studies, speaker models were trained by optimizing (1) or (2) directly on the enrollment data of that speaker. The same optimization criteria would then be used as the similarity score between unknown sample and the given model(s). The current paradigm, however, uses a two-stage training process. First, a *universal background model* (UBM) is trained by pooling a large number feature vectors from different speakers and optimizing (1) or (2) with any suitable clustering algorithm. The UBM serves as *prior* information about the general (speaker-independent) distribution of the spectral feature space, and it is used as a form of regularization (smoothing) in the training. To be precise, for the GMM model the mean vectors of the UBM, $\boldsymbol{\mu}_k^{\mathrm{UBM}}$, are adapted as,

$$\boldsymbol{\mu}_k^{\mathrm{adapted}} = \alpha_k \boldsymbol{E}_k(X) + (1 - \alpha_k)\boldsymbol{\mu}_k^{\mathrm{UBM}}, \tag{3}$$

where

$$\alpha_k = \frac{\sum_{i=1}^{N} P(k|\boldsymbol{x}_i)}{r + \sum_{i=1}^{N} P(k|\boldsymbol{x}_i)} \quad \text{and} \quad \boldsymbol{E}_k(X) = \frac{1}{n_k}\sum_{i=1}^{N} P(k|\boldsymbol{x}_i)\boldsymbol{x}_i. \tag{4}$$

Here $P(k|\boldsymbol{x}_i)$ is the posterior probability of vector $\boldsymbol{x}_i$ originating from the *k*th Gaussian, n_k is the soft count of vectors assigned to the *k*th Gaussian, and *r* is a fixed constant known as *relevance factor*. Typically *r* is a fixed constant (Reynolds et al., 2000) but data-adaptive relevance factor using fuzzy control rule has also been suggested (Juang et al., 2003). In this study, we use fixed constant $r = 16$ as usually done in speaker verification. Note that only the mean vectors are adapted, and the rest of the parameters are shared between speakers. For more details, refer to (Reynolds et al., 2000). For the VQ model, the adaptation formulae are a special case of (3) and (4) with the assumption that $P(k|\boldsymbol{x}_i) = 1$ for the nearest centroid vector in UBM, and $P(k|\boldsymbol{x}_i) = 0$ otherwise. For the VQ adaptation, we use relevance factor $r = 12$ as in (Hautamäki et al., 2008).

In the recognition phase, the average log likelihood (mean square error in the case of VQ) of the data, in respect both to the target speaker and the UBM, are evaluated, and their difference gives the *normalized score*. Normalization with the background model equalizes the score ranges of different speakers and test segments so that a common verification threshold can be used. The normalized score is finally compared with a verification threshold to give the accept/reject decision.

2.2. Clustering algorithms

Optimal algorithms for solving the clustering problem have exponential time complexity (Garey et al., 1982). Thus, all methods for data sets consisting of thousands or millions of training vectors are based on different heuristics; several hundreds of methods have been proposed in literature (Jain et al., 1999). For the comparisons in this paper, we include algorithms that, according to our experience (Fränti and Virmajoki, 2006), consistently provide high quality clustering, and algorithms that are popular due to their simplicity or for other reasons. We include two hierarchical algorithms (PNN, SPLIT) and six iterative algorithms (*K*-means, SOM, RS, SM, FCM, GA). Random clustering is used as reference points. GMM, on the other hand, is the *de facto* standard in text-independent speaker recognition, and provides another good reference point.

Random: A trivial method for modeling the data is to construct the codebook from *K* randomly chosen data vectors. The random codebook will also be used as the initial solution for the iterative algorithms described below, but serves also as a reference solution for measuring the quality of the clustering algorithms. A good clustering algorithm should produce significantly better codebook than the random selection.

Repeated K-means: K-means (McQueen, 1967) starts from any initial solution, which is then iteratively improved by two optimization steps as long as improvement is achieved. The algorithm is known as *Linde-Buzo-Gray* (LBG) or *generalized Lloyd algorithm* (GLA) in vector quantization (Linde et al., 1980). Since *K*-means is sensitive to the initial solution, we apply it repeatedly each time starting from a new random initial solution (Duda and Hart, 1973). The codebook providing the smallest MSE is retained as the final solution.

SOM: *Self-organizing map* (Nasrabadi and Feng, 1988) is a neural network approach to the clustering problem. The neurons in the network are connected with a 1-D or 2-D structure, and they correspond to the code vectors. Each feature vector is fed to the network by finding the nearest code vector. The best matched code vector and its neighboring vectors in the network are updated by moving them towards the input vector. After processing the training set by a predefined number of times, the neighborhood size is shrunk. The entire process is repeated until the neighborhood size shrinks to zero.

PNN: *Pairwise nearest neighbor* (Equitz, 1989; Fränti et al., 2000) generates the codebook hierarchically. It starts by initializing each feature vector as a separate code vector. Two code vectors are merged at each step of the algorithm and the process is repeated until the desired size of the codebook is obtained. The code vectors to be merged are always the ones that results in the least distortion. We use the fast exact PNN algorithm introduced in (Fränti et al., 2000).

SPLIT: An opposite top-down approach starts with a single cluster of all the feature vectors. New clusters are then created one at a time by dividing the existing clusters. The splitting process is repeated until the desired number of clusters is reached. This approach usually requires much less computation than the PNN. We use the algorithm in (Fränti et al., 1997b) that always selects the optimal hyperplane, dividing the particular cluster along its principal axis, augmented with a local repartitioning phase at each division step. This variant gives comparable results to that of the PNN but with much faster algorithm.

SM: *Split-and-Merge* (Kaukoranta et al., 1998) is an iterative algorithm that modifies the codebook by a series of split and merge operations. At every step, the code vectors to be split and merged are chosen as the ones that provide best improvement (split), or least increase (merge) in the distortion. The algorithm provides high quality codebooks but with a significantly more complex implementation than the other algorithms.

RS: *Random swap* algorithm (Fränti and Kivijärvi, 2000) starts with a random codebook, which is then iteratively improved. At every step, a randomly selected code vector is tentatively re-allocated to a randomly chosen training vector. The new candidate codebook is fine-tuned by two iterations of *K*-means, the solution is then evaluated and accepted if it improves the previous solution. The algorithm is iterated for a fixed number of iterations. This trial-and-error approach is much simpler to implement than the split-and-merge algorithm, and is surprisingly effective. It was shown to find the optimal global allocation of the codebook in an expected time of $O(N^2K)$ Fränti et al., 2008. See Fig. 2 for illustration of the algorithm.

FCM: *Fuzzy C-means* (Dunn, 1974) generalizes *K*-means to fuzzy clustering, in which data vectors can belong to several partitions at the same time with a given weight. Traditional *K*-means is then applied in the final step in order to obtain the centroids (codebook) from the fuzzy partitions. Another alternative would be to formulate fuzzy-MAP adaptation based on the fuzzy memberships. Since we are not aware of such formulation, we use the centroids obtained from the hard partitions.

GA: *Genetic algorithm* generates a set of solutions (called population) and evolves it using the survival of the fittest principle. In (Fränti et al., 1997a), PNN is used as the crossover to generate new candidates, which are further fine-tuned by *K*-means. The algorithm has outperformed so far every competitive clustering algorithm for more than a decade already. Slightly better results have been reported only by other (more complicated) GA variants [FräntiShrink2006]. It therefore serves as a good reference point for clustering model.

GMM : We use the *expectation–maximization* (EM) algorithm (Bishop, 2006; McLachlan and Peel, 2001) for training the GMM as described in (Reynolds and Rose, 1995). In the EM algorithm, an initial guess is made for the model parameters, and the solution is then improved by using two optimization steps similar to *K*-means. Since the EM algorithm is also sensitive for the initialization, we apply it repeatedly starting from a new random solution, which is always first fine-tuned by *K*-means. The result providing the highest likelihood is retained as the final model. An important consideration in GMM is the type of the covariance matrices of the Gaussian components. As generally done with MFCC features, we use diagonal covariance matrices instead of full covariances due to numerical and computational reasons: for limited data, full covariance matrices easily become singular (ill-conditioned). Using diagonal covariances is also computationally efficient since no full covariance matrix inversions are required.

2.3. Number of clusters

The number of clusters (model order) is a control parameter of a clustering algorithm which must be optimized for a given application. In recognition applications, this is usually done by picking the model order that gives best recognition accuracy on a development set. Practice has shown that, irrespective of the implementation details, corpus and chosen short-term features, the best accuracy is typically found using $K = 64–2048$ code vectors or Gaussian components. The main drawback of this approach is the expensive

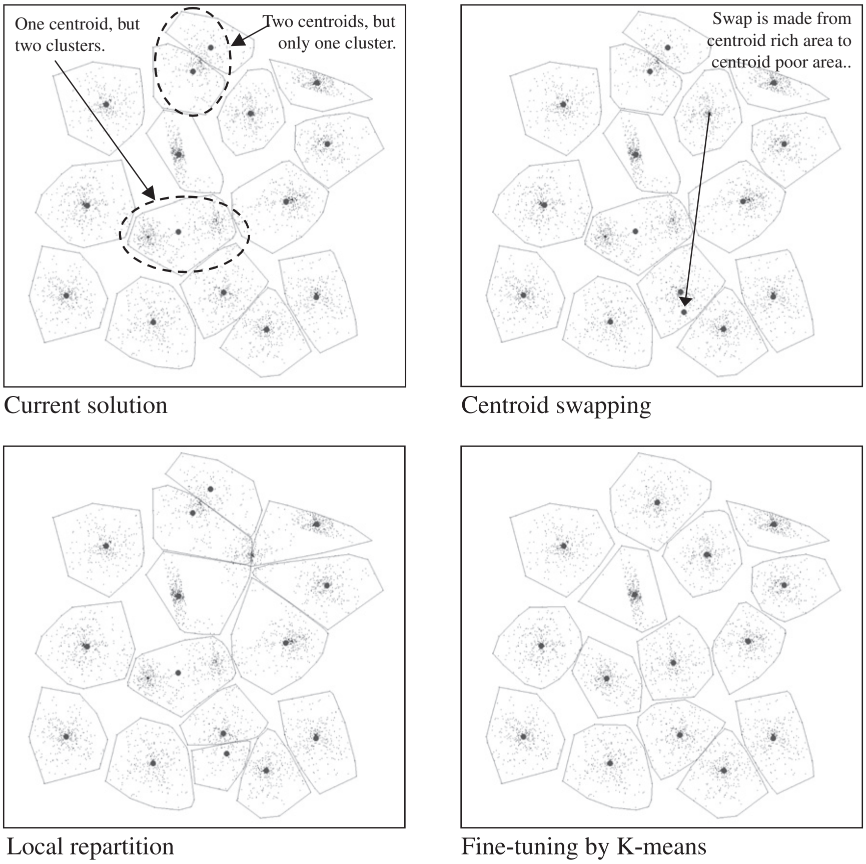
One centroid, but two clusters.
Two centroids, but only one cluster.
Current solution
Swap is made from centroid rich area to centroid poor area..
Centroid swapping
Local repartition
Fine-tuning by K-means

Fig. 2. Illustration of a single step of the random swap (RS) algorithm. A randomly chosen centroid is re-allocated to new location, followed by *K*-means fine-tuning. The new solution is accepted if it provides smaller distortion than the original codebook.

computations involved – for each considered model order, one needs to re-train speaker models (and UBM if MAP adaptation is used) and re-classify the development test samples.

In clustering research, large number of *clustering validity indices* have been proposed for automatically detecting the number of clusters (e.g. Bezdek and Pal, 1998; Davies and Bouldin, 1979; Geva et al., 2000; McLachlan and Peel, 2001; Milligan, 1981; Sarkar et al., 1997). In early phase of this study, we also evaluated the classical *F*-ratio (based on ANOVA test procedure Ito, 1980) and Davis–Bouldin index (DBI) (Davies and Bouldin, 1979) in VQ-based speaker modeling, but found the selected model order to correlate poorly with recognition accuracy. Even though these indices have been reported to work reasonably well for low-dimensional features and data with clear clustering tendency (Kärkkäinen and Fränti, 2002), they are affected by overlapping clusters and noise (e.g. Bezdek and Pal, 1998), as well as increasing dimensionality. Since speech features have tens of dimensions and are unlikely to have any clustering tendency (Kinnunen et al., 2001), this may explain the result. For practitioners, we therefore recommend to use the optimize-on-devset procedure.

3. Experimental setup

3.1. Corpora and features

For the experiments, we use three corpora: TIMIT, NIST 1999 and NIST 2006 as documented in Tables 1 and 2. TIMIT represents laboratory quality speech recorded in highly controlled conditions. It was selected for the purpose of parameter optimization in an initial stage of our study. The NIST 1999 and NIST 2006 corpora, on the other hand, represent uncontrolled, conversational telephone quality speech, which is expected in real applications. We use 12- and 36-dimensional mel-frequency cepstral coefficient (MFCC) features for the NIST 1999 and NIST 2006, respectively (see below).

The TIMIT corpus (Linguistic Data Consortium) consists of 630 speakers, and for each speaker there are 10 speech files. We split the files into non-overlapping training and test sections of 70% (22 s) and 30% (9 s), respectively. For consistency with the NIST files, TIMIT files were anti-alias filtered and downsampled from 16 kHz to 8 kHz.

The NIST 1999 corpus (Martin and Przybocki, 2000) consists of 539 speakers (230 males, 309 females). We use the training section of the corpus for our experiments. Each speaker's training data is given in two files labeled "a" and "b", and each has duration of one minute. We use the "a" files for training and the "b" files for classification. For a given speaker, these two files are from the same telephone number but from two different telephone calls (sessions). Different speakers may have same or different type of handset (electret or carbon button). To evaluate verification performance, we match each of the test files per each of the speakers, yielding a total number of 539 × 539 = 290,521 test trials, of which 539 are genuine and the remaining 289,982 are impostor trials.

From the NIST 2006 corpus, we have selected the common "core condition" as specified in the NIST 2006 SRE evaluation plan (NIST 2006 Speaker Recognition Evaluation Webpage, 2006). This benchmark test consists of 816 target speakers (354 males, 462 females) and a total number of 53,966 verification trials (5077 genuine, 48,889 impostors).

We use the mel-frequency cepstral coefficients (MFCCs) as the acoustic features (Huang et al., 2001), see Table 2. Each frame is multiplied by a 30 ms Hamming window, shifted by 10 ms. From

Table 1
Summary of the speech material.

	TIMIT	NIST 1999	NIST 2006
Language	English	English	Mostly English[a]
Speakers	630	539	816
Test trials	N/A	539 genuine + 289,982 impostor	5077 genuine + 48,889 impostor
Speech type	Read	Conversational	Conversational
Quality	Laboratory	Telephone	Telephone
Sampling rate	8.0 kHz	8.0 kHz	8.0 kHz
Session mismatch	Matched	Mismatched	Mismatched
Channel mismatch	Matched	Mixed	Mismatched
Training data (avg.)	22 s	~1 min	~2.5 min
Test data (avg.)	9 s	~1 min	~2.5 min

[a] Small part of the data contains Arabic, Mandarin, Russian, or Spanish speakers.

Table 2
Evaluation set-up for each corpus. UBM = universal background model, d = feature dimensionality, N = number of vectors.

Corpus	Evaluation task	Features used	UBM training data and UBM type	UBM training vectors
TIMIT	Identif.	MFCC ($d = 12$)	N/A	N/A
NIST 1999	Verif.	MFCC ($d = 12$)	NIST 2000, single UBM	$N = 591.378$
NIST 2006	Verif.	MFCC + $\Delta + \Delta^2$ ($d = 36$)	NIST 2004, gender-dependent UBMs	$N = 500.000$ per gender

the windowed frame, magnitude spectrum using fast Fourier transform (FFT) is computed and then filtered with a bank of 27 triangular filters spaced linearly on the mel-frequency scale. The log-compressed filter outputs are then converted into cepstral coefficients by discrete cosine transform (DCT), and the coefficients 1–12 are retained.

For the TIMIT and NIST 1999 corpus, we use the 12 MFCCs as features, followed by utterance-level mean and variance normalization to give zero mean, unit-variance features. For these two corpora, voice activity detection (VAD) is not needed; TIMIT samples are short and of high-quality, containing mostly speech. The NIST 1999 corpus, in turn, has been pre-processed by NIST for silence removal. This simple setup is sufficient on these data sets according to our experience.

For the more challenging NIST 2006 data, our front-end includes additional *RelAtive SpecTrAl* (RASTA) filtering Hermansky and Morgan, 1994 to mitigate convolutive channel effects, followed by estimation of the Δ and Δ^2 parameters to capture local spectral dynamics. Finally, an adaptive energy-based algorithm is used for picking speech-only frames, followed by utterance-level mean and variance normalization. For the NIST 2006 corpus, voice activity detection is crucial – according to (Hautamäki et al., 2007), error rates may increase near chance level if VAD is excluded. The Matlab code of the energy VAD used in this study is available in (Kinnunen and Li, 2010).

For the NIST 1999 and NIST 2006 corpora, we need universal background models. For the NIST 1999 data, we use a subset of the 1-speaker detection task training files of the NIST 2000 speaker recognition corpus (Linguistic Data Consortium) and for the NIST 2006 data, we use a subset of the 1-side training files of the NIST 2004 SRE corpus. For NIST 1999 we use gender-independent UBM and for NIST 2006 we use separate UBMs for female and male speakers.

3.2. Performance criteria

To assess speaker verification performance, we use the *detection error tradeoff* (DET) curve (Martin et al., 1997) as an evaluation tool. The DET curve presents the trade-off between the two detection error rates, false acceptance rate (FAR) and false rejection rate (FRR), in a normal deviate scale over all decision thresholds. For Gaussian score distributions, the resulting DET curves are straight lines. As an average error measurement, we report the *equal error rates* (EERs), i.e. the error rate corresponding to point FAR = FRR. We also provide the FAR and FRR at a few additional operating points corresponding to security and user-convenient application scenarios.

To measure computational efficiency of background model training, we use the average CPU time over 10 repetitions. All the clustering algorithms have been implemented using either C or C++ languages. The NIST 2006 experiments were carried out in a Dell PE2900 workstation with two 3 GHz X5450 CPUs, 48 GB of RAM and CentOS release 5.3 operating system. Care was taken in excluding the file I/O overhead from the running times. The TIMIT and NIST 1999 experiments were carried out in two older Dell Optiplex G270 computers (2.8 GHz CPU) and 1 GB RAM.

3.3. Parameter setting of the clustering algorithms

The clustering algorithms have several control parameters that should be fixed beforehand. We document the selection of the parameters and comment their importance for the success of the algorithm in the following. Summary is given in Table 4. Note that, even though the number of clusters (K) is a control parameter, it is common for all the algorithms and hence not counted here.

Random: This algorithm has no control parameters.

Repeated K-means: K-means does not have any control parameters but its quality strongly depends on the initial solution. We therefore repeat the algorithm several times (R) by restarting from different random initial solutions as originally proposed in (Duda and Hart, 1973). As a negative side, this also multiplies the processing time R times compared to that of a single run of K-means. Here we set $R = 10$.

SOM: The SOM algorithm does not depend much on the initialization but it is very sensitive to the parameter setup (Fränti, 1999). We fix the initial neighborhood size ($D_{\max}$) to 16, and then study the learning rate (α), and the number of iterations (I) on TIMIT corpus. Based on the identification results in Table 3, we fix the learning rate as $\alpha = 0.01$, and the number of iterations to $I = 1000$.

PNN and SPLIT: The hierarchical algorithms (PNN and SPLIT) have no control parameters and they always produce the same result. The SPLIT approach itself includes several design alternatives such as which cluster to split and how to split it. However, the

Table 3
Dependency of the SOM performance on the control parameters for TIMIT corpus using codebook size = 32. The reported numbers are closed-set identification error rates (IER%) over the whole TIMIT corpus with 630 speakers.

Number of iterations (I)	Learning rate (α)			
	0.001	0.01	0.1	1
5	66.0	15.9	3.2	60.5
10	48.1	10.3	3.2	59.5
20	21.4	5.2	2.9	60.3
50	19.7	2.7	**1.4**	59.2
100	11.4	2.2	2.9	61.4
1000	1.7	**1.4**	4.0	62.2

Table 4
List of control parameters of the algorithms, and the values considered. The selected values are shown in boldface.

Algorithm	Control parameters	Values tested
Random	N/A	N/A
Rep. *K*-means	Number of restarts (R)	R = 5, **10**, 100
SOM	Number of iterations (I)	I = 5, 10, 20, 50, 100, **1000**, 10000
	Maximum learning rate (α)	α = 0.001, **0.01**, 0.1, 1
	Size of the initial neighborhood ($D_{\max}$)	$D_{\max}$ = **16**
PNN	N/A	N/A
SPLIT	N/A	N/A
RS	Number of iterations (I)	I = **2500**, 5000
SM	Number of splits before merging (H)	H = ***K*** (codebook size)
GA	Number of generations (I)	I = until no improvement
	Size of generation (Z)	Z = 10
FCM	Number of FCM iterations (I_{FCM})	I_{FCM} = 200
	Number of *K*-means iterations (I_{km})	I_{km} = **10**, 100
GMM	Number of restarts (R)	R = **10**
	Variance floor ($\sigma^2_{\min}$)	$\sigma^2_{\min} = 1.52 \times 10^{-5}\sigma^2$

proposed solution in (Fränti et al., 1997b) works very well for all data sets without the need of any data-dependent parameter tuning. It was also aimed at maximum quality (at the cost of speed). But since it remains the fastest among the tested algorithms, the faster variants are not considered.

SM: In the SM algorithm, there is one control parameter (step size H), which defines how many subsequent split steps are performed before the same amount of merge operations. We follow the recommendation of Kaukoranta et al. (1998), and fix it to be equal to the size of the codebook ($H = K$). Smaller values would provide slightly higher MSE using less processing time, whereas higher values do not provide much further improvement. The exact choice of this parameter is not critical. The other parameters are insignificant and the default values described in (Kaukoranta et al., 1998) can be used.

RS: In the RS algorithm, we must select the number of iterations (I), which determines the trade-off between processing time and quality. Our previous experience indicates that the number of iterations should be proportional to the number of input vectors (N) to guarantee high quality result. We consider the values I = 2500 and 5000. The first one will be used later as there was not much difference in accuracy when tested with TIMIT.

FCM: We need to fix the number of iterations. According to previous experiments [VirmajokiShrink2006], they are fixed as shown in Table 4.

GA: We need to fix the population size and the number of generations. Their selection is merely a trade-off between quality and time. The results in (Fränti, 2000) have shown that even the faster variant provides very good performance. We therefore fix the generation size to z = 10, accordingly, and iterate the algorithm until no improvement is found (usually 5–20 iterations).

GMM: The initial mean vectors for the expectation–maximization (EM) algorithm are initialized by random selection from the training set, followed by 10 *K*-means iterations. Following this, the covariance matrices are computed from the vectors assigned to each cluster, and the weights are set to the relative count of vectors assigned to the cluster. After initialization, the expectation and maximization steps are iteratively repeated until the relative increase in likelihood falls below a given threshold (ε). Like in K-means, we repeat the algorithm R times, each time starting from a new random solution, and choose the final model as the one which yields the highest likelihood. Here, we fix the parameters as $\varepsilon = 2^{-16}$ and R = 10. We also need to set a *variance floor* ($\sigma^2_{\min}$) for each dimension to prevent components becoming singular (Reynolds and Rose, 1995). The values were optimized on the TIMIT data and fixed to 1.52×10^{-5} times the variance of the training set. The number of restarts (R) and the variance floor are important control parameters of the algorithm, which must be setup experimentally since there are no good theoretical guidelines how to set them optimally for a given data set (Reynolds and Rose, 1995). The selection of the convergence threshold (ε), on the other hand, is much less critical and can be considered a fixed constant.

4. Results and discussion

4.1. Speaker recognition accuracy

First, we present results on the NIST 1999 corpus and study the effect of the background model. We use the random swap (RS) as a representative vector quantization method, and compare it against the Gaussian mixture model (GMM) trained with the repeated expectation–maximization (EM) algorithm. The speaker models are trained independently without UBM and without score normalization (EM, RS), or by using MAP adaptation from the UBM and with UBM score normalization (EM + MAP, RLS + MAP). The DET plots are presented in Fig. 3 and representative score distributions are shown in Fig. 5.

VQ achieves higher accuracy at small FAR levels, but the differences become smaller when UBMs are used. The UBM adaptation and normalization improves the accuracy of both methods. For the rest of the experiments, we use background modeling and thus, the focus is on comparing different clustering methods to train the UBM.

While UBM adaptation and score normalization are known to improve accuracy, it is much less studied how the set-up of UBM training affects recognition accuracy. To study this in more detail, we consider two different methods to initialize GMM: (1) the repeated EM method described in Sections 2.2 and 3.3, and (2) a deterministic method used in (Kinnunen et al., 2009). The latter was specifically optimized on the latest NIST corpora to give good recognition accuracy with small time consumption. It uses splitting algorithm to generate an initial codebook, which is fine-tuned by seven *K*-means iterations; the covariances and weights are then initialized from the codebook partitions; finally, two EM iterations are executed (further EM iterations did not improve accuracy in Kinnunen et al. (2009)). Comparative results with VQ-UBM (using the random swap algorithm) are shown in Fig. 4.

The two alternative methods of training GMM do exhibit different performance. At small FRR levels (lower right corner), the repeated EM clearly outperforms the faster heuristic variant; at small FAR levels (upper left corner), in turn, the order is reversed, even though the difference is smaller. RS algorithm gives the same performance with repeated EM at small FRR levels and model size

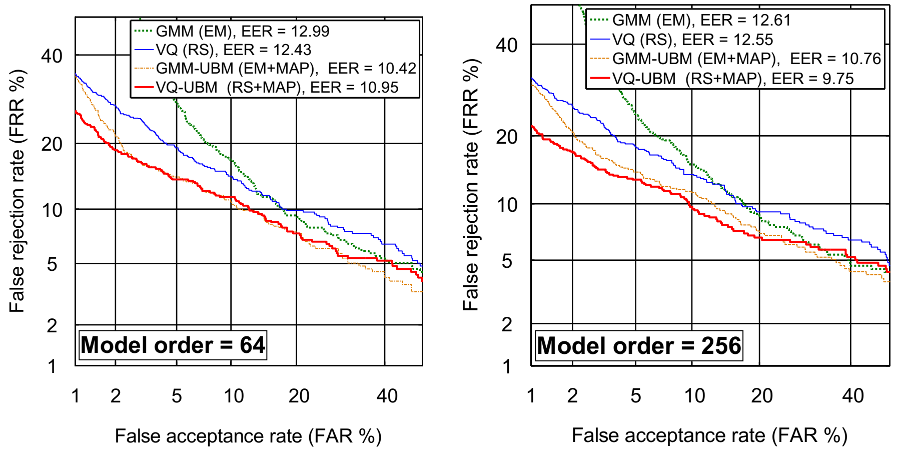

Fig. 3. Results on the NIST 1999 corpus (Model orders $K = 64$ and $K = 256$) using VQ and GMM approaches, with and without UBM.

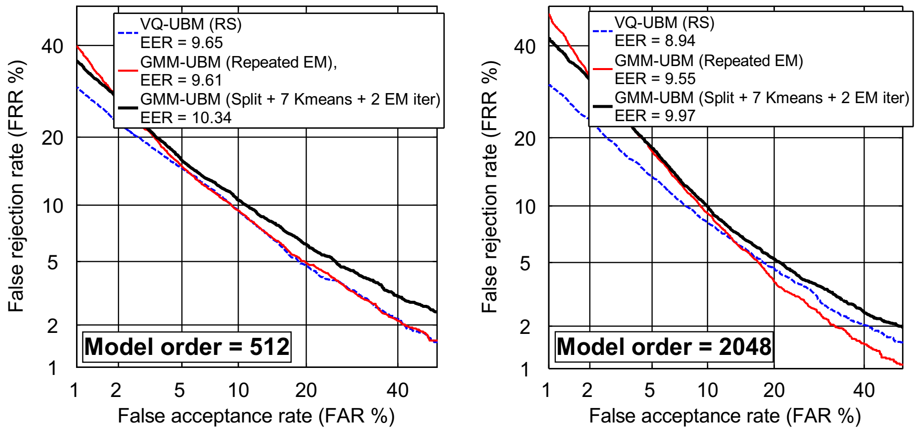

Fig. 4. Comparing random swap (RS) to GMM with two different initializations on the NIST 2006 corpus. The UBMs are trained with the indicated methods.

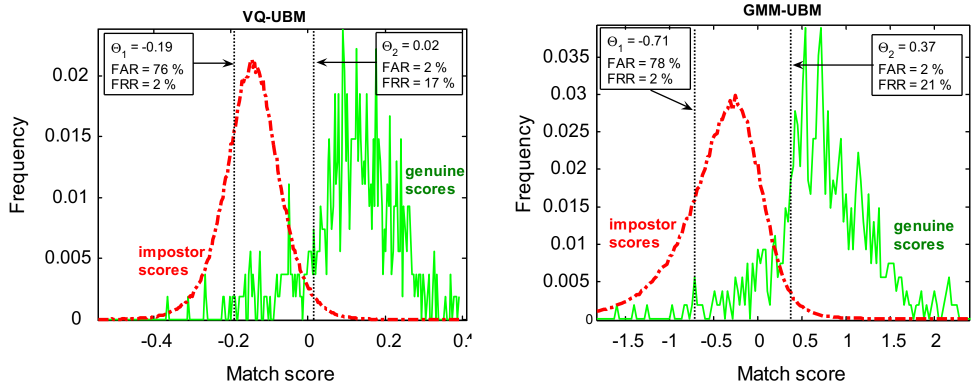

Fig. 5. Match score distributions for VQ-UBM and GMM–UBM on the NIST 1999 corpus for model size 256. Two operating points (thresholds) with the corresponding error rates are indicated.

$K = 512$; however, RS outperforms EM at small FAR levels. This difference is even larger when model order is increased to $K = 2048$. At small FRR levels and with $K = 2048$, repeated EM gives the best performance. In summary, VQ-UBM is better suited for security applications (small FAR desired), whereas GMM–UBM is better for user-convenience applications (small FRR desired).

For more complete analysis, Table 5 summarizes error rates at a few selected operating scenarios for both the NIST 1999 and the NIST 2006 corpora. The notation "FAR @ FRR = 1%" means that the verification threshold has been adjusted to give 1% FRR, and the corresponding FAR is reported in the table; similarly for the other error type. Independent of the corpus and feature set-up, GMM–UBM seems to be better for user-convenience and VQ-UBM for security application, respectively.

McNemar's significance test at 95% confidence level (Huang et al., 2001; Leeuwen et al., 2006) was performed at each operating

Table 5
Error rates for the VQ-UBM and GMM–UBM for two application scenarios. Here d = feature dimensionality, K = model order, *rep.EM* = repeated expectation–maximization (EM), *heur.EM* = Split + 7 K-means initialization, followed by two EM iterations.

Corpus, feature dimensionality (d) and model order (K)		NIST 1999 d = 12, K = 256		NIST 2006 d = 36, K = 2048		
Application scenario	Model and UBM training method	VQ-UBM (RS)	GMM–UBM (rep.EM)	VQ- UBM (RS)	GMM–UBM (heur.EM)	GMM–UBM (rep.EM)
	EER	9.75	10.76	8.94	9.97	9.55
User-convenient application	FAR@FRR = 1%	85.56[a]	80.82	67.51[a]	99.42	52.36
	FAR@FRR = 2%	74.64	74.87	41.01[a]	48.97	32.82
	FAR@FRR = 5%	41.65[a]	36.41	18.09[a]	21.14	17.16
	FAR@FRR = 10%	9.69[a]	12.84	7.73[a]	9.96	9.22
Security application	FRR@FAR = 1%	21.71[a]	30.98	30.75[a]	42.17	48.20
	FRR@FAR = 2%	17.07[a]	20.59	23.30[a]	31.91	32.62
	FRR@FAR = 5%	12.99	14.10	13.69[a]	18.20	17.61
	FRR@FAR = 10%	9.65	11.32	8.29[a]	9.97	9.18

[a] Significantly different from GMM–UBM (rep. EM), at the confidence level of 95%, as evaluated using McNemar's test.

point between GMM–UBM (repeated EM) and VQ-UBM (RS). We measure the difference in the decisions on the impostor trials in case of user-convenience application (FAR @ FRR = 1..10%), difference in the genuine trials in case of secure application (FRR @ FAR = 1..10%), and all trials at the EER operating point. Statistically significant differences are indicated by an asterisk ($*$) in Table 5. On NIST 2006, GMM-UBM outperforms VQ-UBM at the user-convenient scenario, whereas the situation is reversed in the security scenario. In the NIST 1999, the same conclusion holds except in a few cases. In general, the differences are smaller near the EER operating point and only in impostor but not in genuine trials. The reason why differences are not always significant for genuine trials on the NIST 1999 is due to much smaller number of genuine trials in comparison to NIST 2006.

Full comparison of all clustering methods is shown in Fig. 6 on the NIST 2006 corpus. Firstly, all the methods produce significantly better result than random clustering. Secondly, the VQ and GMM methods are equal in terms of EER. Generally VQ methods outperform GMM at small FAR levels (security application), whereas GMM outperforms VQ at small FRR levels (user-convenience applications) when the model order is increased to K = 1024 or 2048. Comparing the VQ methods, the recognition accuracies are very close to each other, in particular for models with large number of clusters. For a smaller number of clusters (K = 16), there are some differences: SOM gives significantly poorer accuracy than the other methods, and the hierarchical methods, PNN and Split, are also slightly poorer. This is hardly significant since the larger codebook sizes are expected to be used in most applications. It must also be noted that, although SOM works reasonably well in these tests, the parameter tuning was a crucial step, which makes it unfavorable method.

An interesting question is whether the clustering quality correlates with increased recognition accuracy. To answer this, we present mean square error (MSE) of the background model in NIST 2006 corpus against the three different error metrics used in Table 5. The results for model order K = 1024 in Fig. 7 suggest that there exists a weak correlation: the PNN, which yields the highest MSE among the tested methods, yields also slightly poorer recognition accuracy. However, the rest of the methods are so close to each other in clustering quality that the differences in recognition accuracy cannot originate from a better clustering algorithm.

4.2. Processing time

In the following examples, we study the computational efficiency of the clustering methods. In the case of iterative algorithms, the processing time increases with the size of the model. In Fig. 8, this can be seen most clearly for K-means and GMM. On the other hand, we use the reduced-search variant of the K-means (Kaukoranta et al., 2000), also in RS, SM and GA, and it exploits the fact that only small portion of the code vectors changes during each iteration. In the case of large codebooks, this proportion becomes smaller and smaller, which makes the dependency on the size of the codebook rather conservative.

In the case of hierarchical algorithms, the processing time depends mainly on the direction of the process. In divisive approach (SPLIT), the algorithm processes from K = 1 to K = N, and therefore, the processing time increases as a function of the codebook size. In the merge-based approach (PNN), the situation is the opposite as it processes from K = N down to K = 1. However, most of the time will be spent in the early stage of the process when there are a large number of code vectors, and therefore, the increase at the smaller sizes does not show anymore in Fig. 8.

The difference of the feature space affects SPLIT and SM methods because they need to calculate principle axis at every splitting stage (SPLIT and SM). These steps have quadratic, $O(d^2)$ time dependency on the dimensionality, and thus, the methods become somewhat slower when the dimensionality increases. All other methods depend linearly on the dimensionality and the size of data, except PNN, which has quadratic, $O(N^2)$, dependency on the size of data.

In summary, the SPLIT method is the fastest of the tested methods. Among the other methods, RS and the Repeated K-means are somewhat slow for the highest codebook sizes, but both of them were tuned here for maximal quality instead of speed.

5. Implementation considerations

A somewhat less appreciated property of an algorithm is the human effort needed to make the algorithm work in practice. There are two viewpoints: (1) *programmer's viewpoint*: complexity of the implementation, and (2) *user's viewpoint*: efforts required to tune up the parameters for a certain corpus. In general, these two factors are difficult to measure quantitatively given the varying skills of programmers and users. To give a rough indication of the first factor, we estimate the number of functions and program code length. Program code length was estimated as the number of lines in the source code, and the file size of the binary code. All programs were compiled using GCC version 4.1.12, without code optimization and debug information. To give an indication of the user effort, we count the number of control parameters.

The numbers in Table 6 match to our own experience in implementing these algorithms. All the programs (except FCM and GMM) consist of the same data structures for the feature vectors, codebooks, partition, and the same functions for distance calculations. In addition to these, K-means includes four functions to

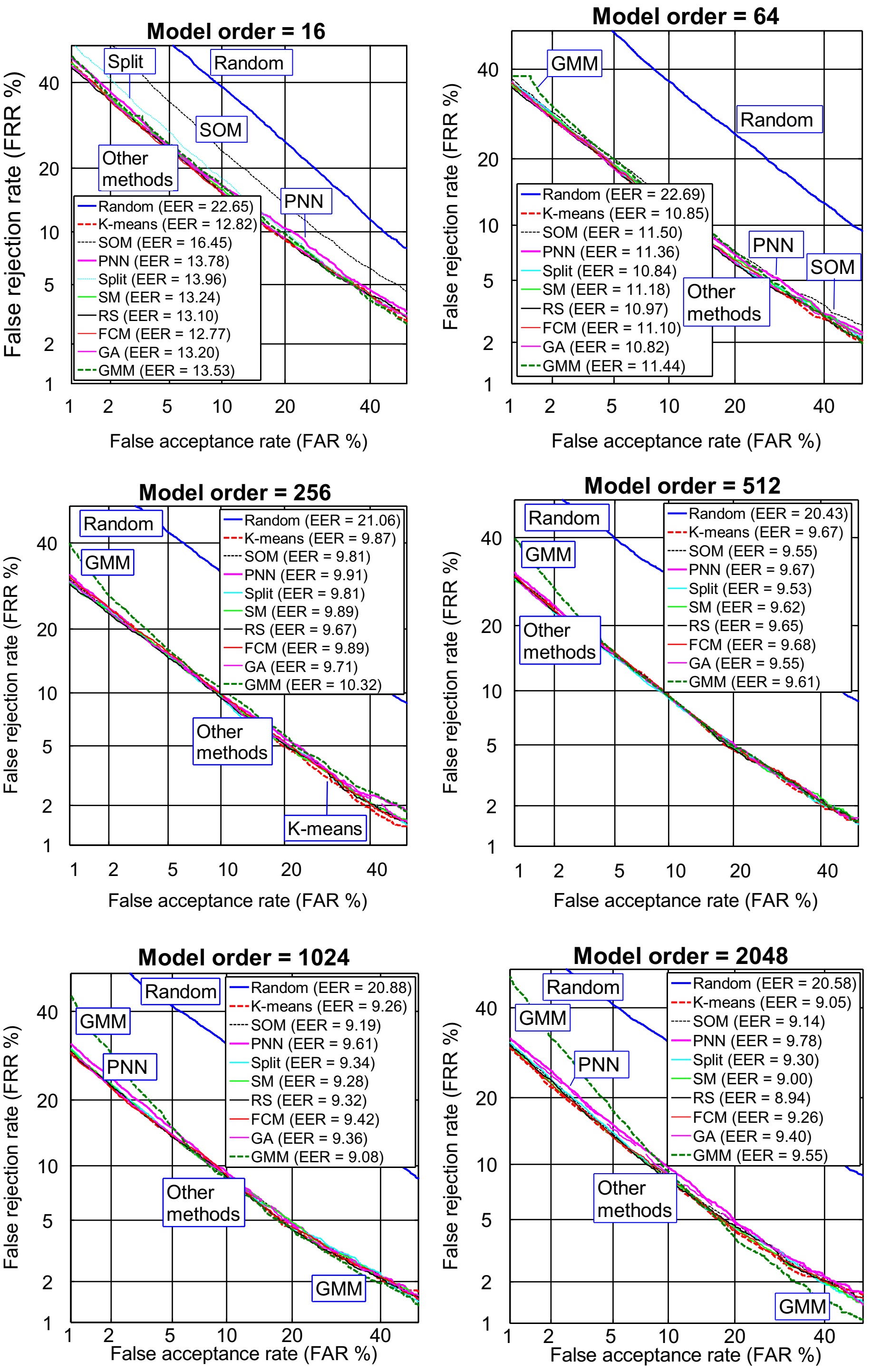

Fig. 6. Recognition results for the NIST 2006 SRE corpus.

generate the partition and centroids for finding the nearest code vector, and for summing up the distortion value. The reduced search variant includes one more. In addition to these, RS includes swap and selection procedures. Thus, their corresponding code sizes are almost the same. In our opinion, these two algorithms are the simplest ones to implement.

The SOM, PNN and FCM algorithms have somewhat longer codes but their implementations are also quite straightforward. The SPLIT, however, is significantly more complex to implement, which is partly reflected in its code size, as well. The algorithm includes functions for calculating the principal component (power method), finding the optimal dividing hyper plane, projection, sorting the data vectors, and a binary tree structure for efficient selection of the next cluster to split, procedure for re-partitioning, and several smaller routines.

Split-and-merge is basically a combination of the PNN and SPLIT codes added with the main routine coordinating between these two steps. However, additional difficulties arise from the fact that updating one of the data structures either by split and merge, will influence the other data structure of the other. As the success of the algorithm requires that both of these steps are implemented accurately, it is far from trivial to make the algorithm work in practice.

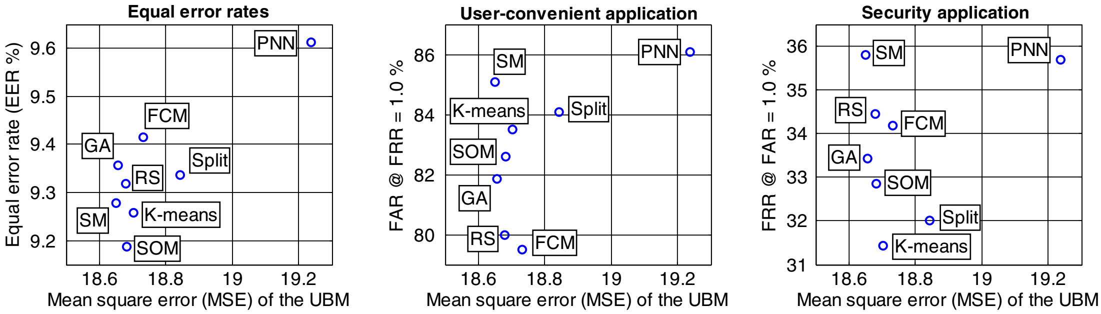

Fig. 7. Clustering quality versus recognition accuracy for model order $K = 1024$. Here quality is measured as the mean square error (MSE) of the (male) universal background model.

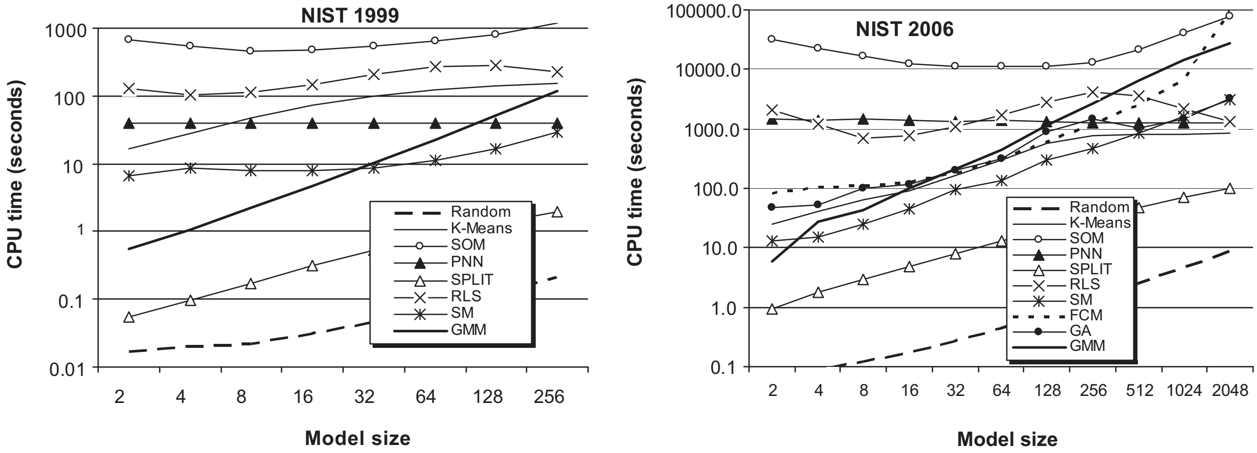

Fig. 8. Running times of the model training for the NIST 1999 (left) and NIST 2006 (right) corpora. The dimensionality of the corresponding features spaces are $d = 12$ and 36. (Processing times of FCM and GA for NIST 1999 are missing due to historical reasons.)

Table 6
Measures estimating the difficulty of implementation.

	Number of control parameters	Number of functions	Lines in source code	Binary code size (bytes)
Random	0	2	26	64,448
Repeated *K*-means	1	5	162	71,821
RS	1	7	226	70,690
SOM	3	7	252	95,448
PNN	0	12	317	107,530
SPLIT	0	22	947	101,778
SM	1	38	1381	131,834
GA	2	21	573	105,956
FCM[a]	2	11	295	128,735
GMM[a]	2	44	1111	87,535

[a] C++ implementation, other methods are in ANSI C.

According to our experience, we cannot really recommend this method for practitioners due to its complex implementation.

GA is combination of PNN and *K*-means algorithm, including a few additional steps in the crossover stage, handling a set of solutions instead of only one, and implementing the selection step. The parameters are rather straightforward to set according to previous recommendations, and the method works rather robust from data set to another. It is to be considered if top performances and the extra (usually insignificant) quality increase in comparison to, say RS or PNN, is desired.

Robust implementation of EM algorithm for GMM requires knowledge of linear algebra and issues related to numerical precision. First of all, in our experience double accuracy should always be used with the EM algorithm. In contrast, standard *K*-means can be implemented even with fixed-point arithmetic on a hand-held device (Saastamoinen et al., 2005). However, in GMM, the covariance matrices can become singular (or non-invertible). The common options are either to limit the variance, or to set the component weight to zero, thus effectively decreasing the model size. Another option is to relocate the component elsewhere with new covariance matrix, or to copy the component from the previous iteration of the EM algorithm. In summary, implementing GMM requires more care with numerics than VQ.

The source codes of the clustering algorithms used in this paper have been made publicly available at http://cs.joensuu.fi/sipu/clustering/.

6. Conclusions

We have presented an extensive comparison of clustering methods in a demanding pattern recognition task including highly noisy telephony speech data. Our main conclusion is that the most important parameter is the order of the model, whereas the choice of the clustering algorithm is less important. It is therefore enough that the data is modeled by any reasonably good clustering algorithm, as long as the size of the codebook or the number of Gaussians is sufficiently large.

We found the choice of the algorithm to be critical only if very small model size is used. However, the result of random clustering indicated that the recognition rate can be significantly high if no clustering is done. We therefore conclude that some clustering algorithm is needed and random sub-sampling is not enough. Regarding the choice of the algorithm, for practitioners we recommend the random swap (RS) algorithm because of its simple implementation and robust performance in all test conditions. If the running time is critical, we recommend the SPLIT algorithm even though its implementation is more complex.

In the current study, all VQ algorithms were optimized for the same squared-error cost function. This explains rather similar results obtained with different VQ variants. For the same reason, since GMM is based on a different objective function, the differences between GMM and VQ type of models tend to be generally larger. In two recent independent studies (Hanilci and Ertas, 2011; Brew and Cunningham, 2010), differences between these two clustering models were reported for different distance functions (Hanilci and Ertas, 2011) and in SVM back-end setting (Brew and Cunningham, 2010). We conclude that training methodology and data selection for UBM (Hasan and Hansen, in press) are worth re-addressing.

According to our tests, GMM–UBM works better for user-convenience applications where false rejections must be minimized, however, the order was reversed in the favor of VQ-UBM when small false acceptances were considered. The observations were similar for both the 12-dimensional MFCC features (NIST 1999 corpus) and the 36-dimensional MFCC + Δ + Δ^2 features (NIST 2006). Differences in the EER region, on the other hand, were not found statistically significant. Interestingly, similar observations have been recently made in another study for NIST 2001; see Fig. 14 in Hanilci and Ertas (2011).

Regarding clustering quality, the results are consistent with the comparisons made with image data (Fränti and Virmajoki, 2006). We expect the results to generalize to other variations of spectral features as well, and to some extent, to other pattern recognition applications.

Acknowledgements

The works of T. Kinnunen and P. Fränti were supported by the Institute for Infocomm Research (I^{2}R) and Nanyang Technological University (NTU) in Singapore, and by the Academy of Finland. The works of I. Sidoroff and M. Tuononen were supported by TEKES under the 4-year PUMS-program. The authors would like to thank Mrs. Amanda Hajnal for carrying out spell-checking on the manuscript.

References

Auckenthaler, R., Carey, M., Lloyd-Thomas, H., 2000. Score normalization for text-independent speaker verification systems. Digital Signal Process. 10, 42–54.

Bagci, U., Erzin, E., 2007. Automatic classification of musical genres using inter-gender similarity. IEEE Signal Process Lett. 14 (8), 521–524.

Bezdek, J., Pal, N., 1998. Some new indices of cluster validity. IEEE Trans. Systems Man Cybernet. Part B 28 (3), 301–315.

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

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Springer.

Brew, A., Cunningham, P., 2010. Vector quantization mappings for speaker verification. In: Proc. 20th Internat. Conf. on Pattern Recognition (ICPR 2010), pp. 560–564.

Burget, L., Matejka, P., Schwarz, P., Glembek, O., Cernocky, J.H., 2007. Analysis of feature extraction and channel compensation in a GMM speaker recognition system. IEEE Trans. Audio Speech Lang. Process. 15 (7), 1979–1986.

Burton, D., 1987. Text-dependent speaker verification using vector quantization source coding. IEEE Trans. Acoust. Speech Signal Process. 35 (2), 133–143.

Campbell, J., 1997. Speaker recognition: A tutorial. Proc. IEEE 85 (9), 1437–1462.

Campbell, W.M., Campbell, J.P., Reynolds, D.A., Singer, E., Torres-Carrasquillo, P.A., 2006a. Support vector machines for speaker and language recognition. Comput. Speech Lang. 20 (2–3), 210–229.

Campbell, W.M., Sturim, D.E., Reynolds, D.A., 2006b. Support vector machines using GMM supervectors for speaker verification. IEEE Signal Process. Lett. 13 (5), 308–311.

Davies, D.L., Bouldin, D.W., 1979. A cluster separation measure. IEEE Trans. Pattern Anal. Machine Intell. 1 (2), 224–227.

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.

Duda, R.O., Hart, P.E., 1973. Pattern Classification and Scene Analysis. John Wiley and Sons, New York.

Dunn, J.C., 1974. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J. Cybernet. 3 (3), 32–57.

Equitz, W.H., 1989. A new vector quantization clustering algorithm. IEEE Trans. Acoust. Speech Signal Process. 37 (10), 1568–1575.

Farrell, K.R., Mammone, R.J., Assaleh, K.T., 1994. Speaker recognition using neural networks and conventional classifiers. IEEE Trans. Speech Audio Process. 2 (1), 194–205.

Fränti, P., Kivijärvi, J., Kaukoranta, T., Nevalainen, O., 1997a. Genetic algorithms for large scale clustering problem. Comput. J. 40 (9), 547–554.

Fränti, P., Kaukoranta, T., Nevalainen, O., 1997b. On the splitting method for vector quantization codebook generation. Opt. Eng. 36 (11), 3043–3051.

Fränti, P., 1999. On the usefulness of self-organizing maps for the clustering problem in vector quantization. In: Proc. 11th Scandinavian Conf. on Image Analysis (SCIA99), Kangerlussuaq, Greenland, pp. 415–422.

Fränti, P., 2000. Genetic algorithm with deterministic crossover for vector quantization. Pattern Recognition Lett. 21 (1), 61–68.

Fränti, P., Kivijärvi, J., 2000. Randomized local search algorithm for the clustering problem. Pattern Anal. Appl. 3 (4), 358–369.

Fränti, P., Kaukoranta, T., Shen, D.-F., Chang, K.-S., 2000. Fast and memory efficient implementation of the exact PNN. IEEE Trans. Image Process. 9 (5), 773–777.

Fränti, P., Virmajoki, O., 2006. Iterative shrinking method for clustering problems. Pattern Recognition 39 (5), 761–765.

Fränti, P., Virmajoki, O., Hautamäki, V., 2008. Probabilistic clustering by random swap algorithm. In: IAPR Internat. Conf. on Pattern Recognition (ICPR'08), Tampa, Florida, USA.

Garey, M.R., Johnson, D.S., Witsenhausen, H.S., 1982. The complexity of the generalized Lloyd-Max problem. IEEE Trans. Inform. Theory 28 (2), 255–256.

Geva, A.B., Steinber, Y., Bruckmair, S., Nahum, G., 2000. A comparison of cluster validity criteria for a mixture of normal distributed data. Pattern Recognition Lett. 21, 511–529.

He, J., Liu, L., Palm, G., 1999. A discriminative training algorithm for VQ-based speaker identification. IEEE Trans. Speech Audio Process. 7 (3), 353–356.

Hanilci, C., Ertas, F., 2011. Comparison of the impact of some Minkowski metrics on VQ/GMM based speaker recognition. Comput. Electr. Eng. 37, 41–56.

Hasan, T., Hansen, J.H.L., 2011. A study on universal background model training in speaker verification. IEEE Trans. Speech, Audio Language Process. in press.

Hautamäki, V., Tuononen, M., Niemi-Laitinen, T., Fränti, P., 2007. Improving speaker verification by periodicity based voice activity detection. In: Proc. 12th Internat. Conf. on Speech and Computer (SPECOM 2007), vol. 2, Moscow, pp. 645–650.

Hautamäki, V., Kinnunen, T., Kärkkäinen, I., Tuononen, M., Saastamoinen, J., Fränti, P., 2008. Maximum a posteriori estimation of centroid model parameters for speaker verification. IEEE Signal Process. Lett. 15, 162–165.

Hermansky, H., Morgan, N., 1994. RASTA processing of speech. IEEE Trans. Speech Audio Process. 2 (4), 578–589.

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

Ito, P.K., 1980. Robustness of ANOVA and MANOVA test procedures. In: Krishnaiah, P.R. (Ed.), Handbook of Statistics 1: Analysis of Variance. North-Holland Publishing Company, pp. 199–236.

Jain, A.K., Murty, M.N., Flynn, P.J., 1999. Data clustering: A review. ACM Comput. Surv. 31 (3), 264–323.

Jain, A.K., 2010. Data clustering: 50 years beyond *K*-means. Pattern Recognition Lett. 31 (8), 651–666.

Juang, Y.-T., Huang, K.-C., Ding, I.-J., 2003. Speaker adaptation based on MAP estimation using fuzzy controller. Pattern Recognition Lett. 24, 2807–2813.

Kaukoranta, T., Fränti, P., Nevalainen, O., 1998. Iterative split-and-merge algorithm for VQ codebook generation. Opt. Eng. 37 (10), 2726–2732.

Kaukoranta, T., Fränti, P., Nevalainen, O., 2000. A fast exact GLA based on code vector activity detection. IEEE Trans. Image Process. 9 (8), 1337–1342.

Kenny, P., Ouellet, P., Dehak, N., Gupta, V., Dumouchel, P., 2008. A study of inter-speaker variability in speaker verification. IEEE Trans. Audio Speech Lang. Process. 16 (5), 980–988.

Kinnunen, T., Li, H., 2010. An overview of text-independent speaker recognition: From features to supervectors. Speech Commun. 52 (1), 12–40.

Kinnunen, T., Kilpeläinen, Fränti, P., 2000. Comparison of clustering algorithms in speaker identification. In: Proc. 28 Internat. Conf. Signal Processing and Communications (SPC 2000), Marbella, Spain, pp. 222–227.

Kinnunen, T., Kärkkäinen, I., Fränti, P., 2001. Is speech data clustered? – Statistical analysis of cepstral features. In: Proc. 7th European Conf. on Speech Communication and Technology, (Eurospeech 2001), vol. 4, Aalborg, Denmark, pp. 2627–2630.

Kinnunen, T., Karpov, E., Fränti, P., 2006. Real-time speaker identification and verification. IEEE Trans. Audio Speech Lang. Process. 14 (1), 277–288.

Kinnunen, T., Saastamoinen, J., Hautamäki, V., Vinni, M., Fränti, P., 2009. Comparative evaluation of maximum a posteriori vector quantization and Gaussian mixture models in speaker verification. Pattern Recognition Lett. 30 (4), 341–347.

Kolano, G., Regel-Brietzmann, P., 1999. Combination of vector quantization and Gaussian mixture models for speaker verification. In: Proc. 6th European Conf. on Speech Communication and Technology (Eurospeech 1999), Budapest, Hungary, pp. 1203–1206.

Kärkkäinen, I., Fränti, P., 2002. Stepwise algorithm for finding unknown number of clusters. In: Proc. Advanced Concepts for Intelligent Vision Systems (ACIVS'2002), Gent, Belgium, pp. 136–143.

Lee, K.A., You, C., Li, H., Kinnunen, T., Zhu, D., 2008. Characterizing speech utterances for speaker verification with sequence kernel SVM. In: Proc. Interspeech 2008, Brisbane, Australia, pp. 1397–1400.

Lei, Z., Yang, Y., Wu, Z., 2005. Mixture of support vector machines for text-independent speaker recognition. In: Proc. 9th European Conf. on Speech Communication and Technology (Interspeech'2005), pp. 2041–2044.

Leeuwen, D.A.v., Martin, A.F., Przybocki, M.A., Bouten, J.S., 2006. NIST and NFI-TNO evaluations of automatic speaker recognition. Comput. Speech Lang. 20, 128–158.

Linde, Y., Buzo, A., Gray, R.M., 1980. An algorithm for vector quantizer design. IEEE Trans. Commun. 28 (1), 84–95.

Linguistic Data Consortium. <http://www.ldc.upenn.edu/>.

Louradour, J., Daoudi, K., 2005. SVM speaker verification using a new sequence kernel. In: Proc. 13th European Conf. on Signal Processing (EUSIPCO'2005), Antalya, Turkey.

Martin, A., Doddington, G., Kamm, T., Ordowski, M., Przybocki, M., 1997. The DET curve in assessment of detection task performance. In: Proc. 5th European Conf. on Speech Communication and Technology, (Eurospeech 1997), Rhodes, Greece, pp. 1895–1898.

Martin, A., Przybocki, M., 2000. The NIST 1999 speaker recognition evaluation – An overview. Digital Signal Process. 10, 1–18.

McLachlan, G., Peel, D., 2001. Finite Mixture Models. John Wiley & Sons, Brisbane.

McQueen, J.B., 1967. Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297.

Meilă, M., Heckerman, D., 2001. An experimental comparison of model-based clustering methods. Machine Learn. 42, 9–29.

Milligan, G.W., 1981. A Monte Carlo study of thirty internal criterion measures for cluster analysis. Psychometrika 46 (2), 187–199.

Nasrabadi, N.M., Feng, Y., 1988. Vector quantization of images based upon the Kohonen self-organization feature maps. Neural Networks 1, 518.

NIST 2006 Speaker Recognition Evaluation Webpage, 2006. <http://www.itl.nist.gov/iad/mig/tests/sre/2006/index.html>.

Pelecanos, J., Myers, S., Sridharan, S., Chandran, V., 2000. Vector quantization based Gaussian modeling for speaker verification. In: Proc. 15th Internat. Conf. on Pattern Recognition (ICPR'2000), vol. 3, pp. 294–297.

Ramachandran, R.P., Farrell, K.R., Ramachandran, R., Mammone, R.J., 2002. Speaker recognition – General classifier approaches and data fusion methods. Pattern Recognition 35 (12), 2801–2821.

Reynolds, D.A., Rose, R.C., 1995. Robust text-independent speaker identification using Gaussian mixture speaker models. IEEE Trans. Speech Audio Process. 3 (1), 72–83.

Reynolds, D.A., Quatieri, T.F., Dunn, R.B., 2000. Speaker verification using adapted Gaussian mixture models. Digital Signal Process. 10 (1), 19–41.

Reynolds, D.A., Campbell, W., Gleason, T., Quillen, C., Sturim, D., Torres-Carrasquillo, P., Adami, A., 2005. The 2004 MIT Lincoln Laboratory Speaker recognition system. In: Proc. Internat. Conf. Acoustics, Speech, and Signal Processing (ICASSP 2005), Vol. 1, 2005, pp. 177–180.

Roch, M., 2006. Gaussian-selection-based non-optimal search for speaker identification. Speech Commun. 48, 85–95.

Saastamoinen, J., Karpov, E., Hautamäki, V., Fränti, P., 2005. Accuracy of MFCC based speaker recognition in Series 60 device. EURASIP J. Appl. Signal Process. 17, 2816–2827.

Sarkar, M., Yegnanarayana, B., Khemani, D., 1997. A clustering algorithm using an evolutionary programming-based approach. Pattern Recognition Lett. 18 (10), 975–986.

Singh, G., Panda, A., Bhattacharyya, S., Srikanthan, T., 2003. Vector quantization techniques for GMM based speaker verification, In: Proc. Internat. Conf. Acoustics, Speech, and Signal Processing (ICASSP'2003), Vol. 2, April 2003, pp. 65–68.

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

Stapert, R.,Mason, J.S., 2001. Speaker recognition and the acoustic speech space. In: Proc. 2001: A Speaker Odyssey – The Speaker Recognition Workshop, pp. 195–199.

Steinbach, M., Karypis, G., Kumar, V., 2000. A comparison of document clustering techniques. In: Proc. KDD Workshop on Text Mining.

Theodoridis, S., Koutroumbas, K., 2009. Pattern Recognition, 4th ed. Elsevier Inc..

Tran, D.T., 2000. Fuzzy Approaches to Speech and Speaker Recognition, Ph.D. Thesis, University of Canberra, Australia, May 2000, p. 154.

Tran, D., Le, T.V., Wagner, M., 1998. Fuzzy Gaussian mixture models for speaker recognition. In: Proc. Internat. Conf. Spoken Language Processing (ICSLP 1998), paper 0798, Sydney, Australia.

Tran, D., Wagner, M., 2002. Fuzzy C-Means Clustering-based speaker verification. In: Proc. Advances in Soft Computing (AFSS 2002), Calcutta, India, 2002, pp. 318–324.

Um, I.-T., Ra, J.-H., Kim, M.-H., 2000. Comparison of clustering methods for MLP-based speaker verification. In: Proc. 15th Internat. Conf. on Pattern Recognition (ICPR'2000), vol. 2, 2000, pp. 475–478.

Wu, D., Li, J., Wu, H., 2009. α-Gaussian mixture modelling for speaker recognition. Pattern Recognition Lett. 30, 589–594.

Yegnanarayana, B., Kishore, S.P., 2002. AANN: An alternative to GMM for pattern recognition. Neural Networks 15, 459–469.