Real-Time Speaker Identification System

BASHAR AL-SHBOUL, HAMAD ALSAWALQAH, DONGMAN LEE
Information and Communication University (ICU)
Daejeon, Yuseong-Gu, 119 Munji-Dong, 305-732
REPUBLIC OF KOREA

Abstract: - Speaker identification applications are the highly commercialized during the speaker recognition and voice biometrics applications. Like other applications it has been experiencing an increasing market and investor interest [1]. However, it is still an open area for research. In this paper we considered creating a real-time speaker identification system as an optimization problem. However, many researches proceeded in this direction. Here, we took a newly proposed technique proposed by Karpov [2] to compare our technique with it. The results, when testing both systems on ELSDSR voices database, show that the newly proposed technique is better than the old proposed one in terms of timing, memory usage and accuracy.

Key-Words:- Speaker Identification, FFCM, Vector Quantization, Fuzzy Clustering

1 Introduction

Speaker Identification (SID) is a process of recognizing who is speaking on the basis of individual information included in speech waves [3]. It is done by comparing the speaker speech waves with others stored in the database in order to identify the closest match of speeches. The input of the system is the speech data and the output is the identified speaker information like number, name, image, etc. Speaker Identification applications include access control and telephone banking. It is reported that high-tech computer thieves annually steal multi billion dollars in the US. Automatic Speaker Recognition (ASR) can substantially reduce this crime by reducing these fraudulent transactions [6].

In this paper, we focus on an optimization problem. The main purpose is to increase the speed of SID. We exploit FFCM [3] to increase accuracy and decrease distortion while existing systems [2] use Vector Quantization as a clustering technique in the pattern matching step. FFCM performs better in terms of time than VQ [3] because it excludes points unnecessary in the calculation step in each iteration while VQ considers all points. Since FFCM does not fall in local minimas as other gradient descent algorithms, similar to VQ, it finds the best cluster for each group and does not stop before that.

Many versions of Speaker Pruning have been proposed for matching and decision making [13, 14]. Pre-Quantization Pruning was used with Cohort Scoring [15, 16]. Users' Pruning algorithms proposed earlier need the time to detect speech tokens and compare with the Speech database to prune users one by one which means that it is a time taking process. Instead, we use P-Tree search to reduce the time for comparing codebooks for all sounds linearly rather than making too many iterations to prune users one by one. This search compares by group cluster, if it fails, it compares group by group, and the best cluster with acceptable error is being chosen.

The performance results show that the proposed scheme increases accuracy as well as performance and shows better results than Karpov's scheme on the test environment.

This paper is organized as follows. Section 2 describes the related work. Section 3 explains design considerations and Section 4 provides the details fo the proposed scheme. Performance

evaluation is discussed in Section 5. Conclusion follows in Section 6.

2 Related Work

A large number of techniques have been proposed for SID systems, but with different modeling algorithms. [2, 4, 5, 6, 7] While some of them use *Neural Networks* (ANN),[12, 18] others [2, 4, 5, 6] exploit clustering techniques such as VQ. And because VQ is known to be faster than ANN in terms of training and does not require empirical parameters' definition such as network structure (topology), it is preferred to be used in real-time systems, unlike ANN where it needs long training times and predefinition for network structure and layers.[17]

Karprov's approach contains two major components that need to be improved: the clustering part, and the matching part. It was chosen because it is the most recent, and the most efficient among previously proposed real-time speaker identification systems.

As for clustering, VQ is used in most speech applications that needs clustering, but looking from another side will show how VQ is time consuming, compared with FFCM, and this is concluded from the different ways of calculations for them. VQ is a statistical approach that includes all data points in clustering calculations regardless of their position on the multi-dimensional space as outliers or they are suitable to represent the data points or not. Another drawback for VQ is that it falls simply in local minima, and thus no optimal solution can be found.

Fuzzy C-Means (FCM) is a data clustering technique wherein each data point belongs to a cluster to some degree that is specified by a membership grade. This technique was originally introduced by Jim Bezdek in 1981 [19] as an improvement on earlier clustering methods. It provides a method that shows how to group data points that populate some multidimensional space into a specific number of different clusters.

FFCM [3] is an extended algorithm of FCM. FCM is an unsupervised clustering technique that finds representatives of the multi-dimensional data based on similarity. Similarity measure, which is widely used, is Euclidean Distance. Membership or relativity of each point to its centroid has a fuzzy value. Higher membership means more similar points. FCM is a learner technique that needs more iterations to find almost specific centers with low error rate. Error rate is calculated by the mean square error.

As for the matching part, the previous model uses Pre-Quentization Pruning (PQP) which is a merged technique between Static Pruning (SP) and Pre-Quentization (PQ) algorithms. SP idea is to maintain an ordered list of the best matching speakers. At each iteration of comparison, M vectors of the clustered speech are inserted for comparison reasons. Then, the worst K matching sounds in the database will be removed from the comparison at the next step. During this process, scoring is being updated cumulatively and efficiently.

3 Proposed Scheme

In this paper, we focus on building a similar system to the one proposed by Karpov [3] and apply our modifications to get better performance and accuracy. The first modification is to apply FFCM instead of VQ in the same system. The second modification is to use P-tree search instead linear search for matching. All the work is done in the matching and searching steps, and modifies clustering and searching techniques in different ways.

FFCM solves the problems of VQ described above. First, it includes only specific points based on their relativity to a centroid position. Second, it does not fall in local minima. Third, it finds better centroids than VQ, and saves time by reducing the number of iterations needed for calculations using a good initialization technique.

3.1 FFCM Work (1st Modification)

First part of modifications is to replace VQ by FFCM. In this part, where the applied algorithm must cluster the received dataset, FFCM does exactly the same as VQ but with more speed. Both algorithms take the dataset, received from the MFCC step, and consider the clustering dataset. The acceleration happens because of some modifications to the basic algorithm, FCM, to be faster. These modifications are in the initialization part and the calculations part. In the initialization part, the initial points to start clustering are no more selected as random unreal points; instead staring points are randomly selected from within the dataset. In the calculations part, the regular FCM is allowed to run five iterations only to allow the algorithm to approximate the final cluster centers, then points will be entered in the calculations depending on their membership values. In [3] the suitable membership value was 0.7 to consider a point in its cluster calculations, and this value was fixed for all our experiments. The stopping criterion was to calculate the same cluster twice with an error of 10^{-5} which includes that the algorithm has converged to an exact cluster center. After running FFCM on the features dataset a clustered features' set will be sent to the matching step or to the enrollment part.

3.2 Matching Work (2nd Modification)

The proposed P-Tree search solves the problem of PQP identified by the cumulative scoring methodology and the multi-iteration users pruning. P-Tree search calculates scoring depending on the distance in a multi-dimensional space and conducts cluster-by-cluster search.

The aim of using P-Tree search is to increase the matching speed by changing the database search technique. Originally, after running FFCM in the training part, clusters must be shaped as a tree. After clustering the results, each speaker should have his own cluster centers. In the old technique, the whole speaker features must be matched with the whole database more than once in order to prune speakers' tree until one speaker is found. The proposed solution reduces this overhead by conducting comparisons cluster by cluster. If no match is found within an error margin, the users will be compared one by one. The search methodology is similar to the Breadth-First Search.

4 Performance Evaluation

For evaluation, we use the ELSDSR database. The language of the DB is English, and it contains two parts, training part which contains 161 speeches for 23 persons (7 for each), and the testing part which contains totally different 46 speeches for the 23 persons (2 for each). Sampling rate is 8 KHz for all sounds. Speakers in the DB are 10 Females, 13 Males, ages covered from 24 to 63. All are Danish except one international student. Each person is given seven different sentences each of which is two lines to train the system. However, two more different paragraphs are used for testing sounds.

As shown in Figure 1, the evaluation results show that using FFCM in the matching step gives better results than using VQ for 128 clusters for each person's speech. For a small number of clusters, VQ shows better results but not as accurate as using FFCM in 128 clusters. It is because FFCM identify users using the small error margin accepted for each user (10^{-5}). Rather than committing a false acceptance, it commits a false reject, which is considered more secure. As for 128 clusters, it is considered enough data to take a correct decision. We repeat the tests for 15 times. 98% accuracy is achieved only once and 100% for the other 14 times.

It is apparent that the more clusters used, the more accurate results. It is because the representatives are few numbers of points which help saving more information in the codebook, but the more space to save is needed also.

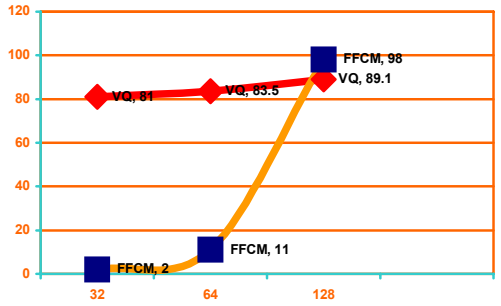

Fig.1: FFCM Vs VQ accuracy results

Figure 2 shows the difference in average time needed for identification for each new user. FFCM runs faster than VQ in terms of time for all clusters sizes because it considers a decreasing number of points in the calculation process.

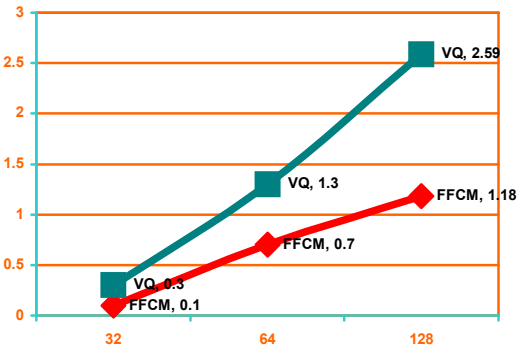

Fig.2: FFCM Vs VQ average time used for identification for each user

Table 1 show the overall results obtained after running VQ and FFCM in terms of accuracy (Acc), error rate (ER), failure rate (FR) and Average Identification Time (AIT).

Table 1: Results after testing FFCM and VQ

	VQ (128)	FFCM (32)	FFCM (64)	FFCM (128)
Acc (%)	89.1	2	11	98-100
ER (%)	10.9	60	19.5	0-2
FR (%)	0	38	69.5	0
AIT(Sec)	2.6	0.1	0.7	1.18

The table shows a comparison of results obtained. The small number under each algorithm name indicates the number of clusters considered.

The smallest number of clusters the less calculations needed, and the less time also. The table also shows that FFCM increases accuracy with the increasing number of clusters with decreasing error rate.

For the first glance you might think that failure rate and error rate are the same, but the most designers prefer to distinguish both error and failure to show some efficiency matters.

Considering the space required for database storage, and taking into an example the first sound in the training part, the sound first extracted into features takes around 2126x20 doubles. However, the data used in calculations are only 500x20. After clustering, only 128x20 is stored for each speech. With simple calculation, 2126/128=17 times less storage space is required. This depends on the file size so 10 times (48MB / 4.7MB) less in the real world. This simple calculation shows that using the proposed scheme will reduce the space needed for speech database at least 10 times.

5 Conclusions

We present a new real-time identification system that is faster than the old proposed one. It is more accurate algorithm in terms of identification, it compresses the size of stored data in the DB at least eight times, and it is at least two times faster.

It is recommended for future work to add a silent detection algorithm to detect sounds at run-time. Because we test on a database silent detection is not needed. To make a compilation to MATLAB code to be compatible with .NET environment in order to implement the whole real project is also a missing part in our project.

References:

[1] Markowitz, Speaker Verification: A survey, Biometric Technology Today, 2001.

[2] Kinnunen, Karpov and Franti, Real-Time Speaker Identification and Verification, IEEE Trans. On Audio, Speech and Language Processing, Vol.14, No.1, 2006.
[3] Al-Zoubi, Hudaib and Al-Shboul, A Proposed Fast Fuzzy C-Means, WSEAS Conf., Corfu, 2007.
[4] Kinnunen, Kilpelainen and Franti, Comparison of Clustering Algorithms in Speaker Identification, Proc. IASTED Int. Conf. Signal Processing and Communications (SPC'00), 2000, pp. 222-227.
[5] Cordella, Foggia, Sansone and Vento, A Real-Time Text-Independent Speaker Identification System, Proc. of the 12th Int. Conf. on Image Analysis and Processing, IEEE, ICIAP'03, 2003.
[6] Campbell, Speaker Recognition: A Tutorial, Proc. of the IEEE, Vol.85, No.9, Sept 1997, pp. 1437-1462
[7] Pusateri and Hazen, Rapid Speaker Adaptation Using Speaker Clustering, In Proc. 7th Inter. Conf. on Spoken Language Processing, Denver, Colorado, 2002, pp. 61-64.
[8] Linde, Buzo, and Gray, An Algorithm for Vector Quantizer Design, IEEE Trans. On Communications, Vol.28, No.1, 1980.
[9] Karpov, Real-Time Speaker Identification, MSc Thesis, University of Joensuu, 2003.
[10] Sun, Liu and Zhong, Hierarchical Speaker Identification Using Speaker Clustering, Beijing Institute of Technology, 2003.
[11] Dempster, Laird and Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, Journal of the Royal Statistical Society, Vol.39, No.1, 1997, pp. 1-38.
[12] Bishop, Neural Networks for Pattern Recognition. Oxford Clarendon Press, 1995.
[13] Pan, Kotani and Ohmi, An on-line hierarchical method of speaker identification for large population, in proc. NORSIG, Kolmarden, Sweden, 2000.
[14] Kinnunen, Karpov and Franti, A speaker pruning algorithm for real-time speaker identification, in Proc. Audio and Video-Based Biometric Authentication (AVBPA), Guildford, UK, 2003, pp. 639-646.
[15] Finan, Sapeluk and Damper, Impostor cohort selection for score normalization in speaker verification, Pattern Recognition. Lett., Vol.18, 1997, pp.881-888.
[16] Ariyaeeinia and Sivakumaran, Analysis and comparison of score normalization methods for text independent speaker verification, in Proc. 5th Eur. Conf. Speech Communication and Technology (Eurospeech), Rhodes, Greece, 1997, pp.1379-1382.
[17] Han and Kamber, Data Mining Concepts and Techniques, 2nd edition, Morgan Kaufmann, Chapter 7, 2006, pp.20.
[18] Pawar, Kajave and Mali, Speaker Identification using Neural Networks, Transactions On Engineering, Computing And Technology, Vol.7, August 2005.
[19] Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, New York, 1981.