SPEAKER INDEPENDENT DISCRIMINANT FEATURE EXTRACTION FOR ACOUSTIC PATTERN-MATCHING

Xavier Anguera

Telefonica Research
Torre Telefonica-Diagonal 00
08019 Barcelona, Spain
xanguera@tid.es

ABSTRACT

Acoustic pattern-matching algorithms have recently become prominent again for automatically processing speech utterances where no prior knowledge of the spoken language is required. Applications of such technology include, but are not limited to, query-by-example search, spoken term detection and automatic word discovery. Obtaining content-aware acoustic features as independent as possible from speaker and acoustic environment variations is a key step in these algorithms. Currently, GMM posteriorgrams are found to outperform the standard MFCC features even though they were not designed to optimize the discrimination between acoustic classes. In this paper we combine the K-means clustering algorothm with the GMM posteriorgrams front-end to obtain more discriminant features. Results on a query-by-example task show that the proposed approaches outperform standard MFCC features by 7.8% absolute P@N and GMM-based posteriorgram features by 3.7% absolute P@N when using a 64-dimensional feature vector.

Index Terms— Pattern-matching, word discovery, query-by-example, k-means

1. INTRODUCTION

Acoustic pattern-matching algorithms were very prominently used in speech processing in the 70's and 80's (see for example [1]) up until HMM-based approaches became more fashionable for being less computationally demanding and obtaining better results. This was made apparent in speech recognition where compact acoustic models could be trained instead of performing a comparison with multiple individual patterns. As time has passed, in order to train acoustic models with growing complexity, the need for manually transcribed and aligned corpora has increased, making development of systems for low-resourced languages more difficult (i.e. languages for which no –or limited– amounts of transcribed audio is available).

Recently we started seeing a reappearance of pattern-matching based applications since they require little (or none) a priori knowledge on the language being processed. Such characteristic is very important when dealing with such low-resource languages, or with particular acoustic conditions for which there is no available data for training or adapting the acoustic models. Furthermore, these algorithms have also benefited from the exponential increase of machine processing power and the proposal of several algorithms for fast matching and for improved search of matching patterns in speech [2, 3, 4, 5].

Regardless of the application, the acoustic front-end used to obtain acoustic features is of big importance for pattern-matching. Such features should represent the content in the audio while being independent of the speaker, the acoustic conditions or the device used for recording. Although traditionally standard speech recognition features (e.g. MFCC) have been used for this task [2], the introduction of posteriorgram vectors [6, 4, 7] has been an important step forward. Given a standard input feature vector, posteriorgram features are obtained either from the phoneme posterior probabilities of a phonetic recognizer [6] or from the posterior probability of each Gaussian in a Gaussian Mixture Model (GMM), trained on similar data regarding the task at hand [7]. Unfortunately, phoneme posteriors have been found to underperform when the language they have been trained on differs from that for which they are used [4]. In addition, as explained below, we do not consider the GMM models, trained using Maximum-Likelihood Expectation-Maximization (EM-ML) to be optimal either, as they do not emphasize the discriminant information between acoustic classes in the training data.

To address this problem, in this paper we use the well-known K-means clustering algorithm to obtain more discriminant feature vectors. We take advantage of the fact that in the K-means algorithm frames are hard-assigned to their closest cluster in order to define clearly bounded clusters. We then use the data assigned to each cluster to train Gaussian mixtures, thus obtaining a final model where Gaussians are in much less overlap than those obtained from EM-ML training. In this paper we propose three possible implementations of this technique. In the first one, we use the Hierarchical K-means algorithm to obtain discriminant clusters defined according to frame density. A second algorithm adapts the well used GMM-posteriorgram model by performing final K-means iteration at the end, which reduces the overlap between Gaussians. Finally, a third algorithm can be implemented as a post-processing to either of the previous two algorithms and performs a binarization of the feature vectors. By doing this we obtain big saving in terms of storage space, although with some performance penalty. All three front-ends have been tested in a query-by-example task on French broadcast news data and obtain interesting improvements over using MFCC parameters and standard GMM models.

2. SPEAKER INDEPENDENT FEATURE EXTRACTION

The algorithms proposed for feature extraction of speaker independent features are built upon the GMM-posteriorgram approach initially applied in pattern-matching in [7]. For a given input feature vector, usually represented as a multidimensional MFCC vector, we can consider the GMM model as performing a change of basis of such data into a different representation, usually higher-dimensional, where the speaker variability is reduced in favor of content discriminancy. Alternatively, we can consider the GMM model as a way to subdivide the multidimensional acoustic space into a finite number of regions, whose distance to every acoustic frame is then used as its representation. In fact, phoneme posteriorgrams are based on the same principle, except that in that case some prior knowledge of a particular language is used in order to define the regions in the acoustic space in which acoustic frames for each phoneme are to be found.

Considering that no transcription is available (therefore we can not train phonetic models with it) we need to automatically divide the acoustic space into acoustically-homogeneous regions. The optimal segmentation of the space should comply with the following two properties. On the one hand, it should be able to cluster the space according to the density of feature points, i.e. more acoustic regions should be defined in areas with higher density of acoustic data, where the distance between acoustically dissimilar regions. Such behavior is important in order to obtain feature vectors where the entropy in each of the dimensions is minimized (i.e. the variance of their values is maximized). On the other hand, the resulting regions should be well bounded and clearly separate acoustically dissimilar regions in order to obtain feature vectors where only one or a few dimensions obtain high values given every input feature.

2.1. EM-ML training versus Hierarchical K-means training

The standard GMM-posteriorgrams approach trains a GMM model via Maximum-Likelihood Expectation-Maximization (EM-ML) using acoustically similar development data. EM-ML training obtains models that fit the given training data by emphasizing the most (e.g. more Gaussian centers) areas with a denser number of feature vectors. Although the EM training approach is able to maximize the likelihood of the data given the model, it tends to generate Gaussian mixtures with high overlap, thus not creating a clear boundary between acoustically dissimilar regions. In addition, after training some of the Gaussians end up with a very small variance as they focus on modeling particular acoustic events. This can cause the posterior probabilities obtained from the Gaussian mixtures to be highly unstable given small data fluctuations like those resulting from speaker variability in producing acoustic sounds.

Alternatively, the K-means algorithm is able to minimize the quadratic mean error by assigning each acoustic frame to the cluster with closest mean according to some defined metric (we use the euclidean distance in this work). By doing so it obtains clearly bounded clusters, which are less prone to error when assigning a given input feature vector to the closest cluster. The k-means algorithm does not find by itself the initial cluster centers, therefore it does not adapt to the density of points in the acoustic space. Fortunately, in acoustic modeling, the K-means algorithm is usually applied in conjunction with a hierarchical approach in order to initialize the Gaussian means in GMM models before the EM-ML algorithm is applied. Hierarchical K-means clustering performs an iterative splitting of the data from 1 to the requested number of final clusters. In our implementation the split is done by slightly displacing each cluster center according to the variance in each dimension. In every step of the iteration we perform K-means until convergence. The result of the algorithm is a set of clearly bounded clusters whose centers are defined taking into account the input data, thus placing more clusters where data is denser.

To illustrate the difference between the result of EMML training and Hierarchical K-means clustering, Figure 1 shows a visualization of the result of modeling a set of 2-dimensional feature vectors by using an EM-ML algorithm and hierarchical K-means. While the EM-ML algorithm positions many Gaussians on top of each other to optimize the likelihood of the model given the data, the Hierarchical K-means algorithm clearly defines the regions in space where each cluster is placed, which accounts for much better defined regions from which we expect to obtain more discriminant feature vectors.

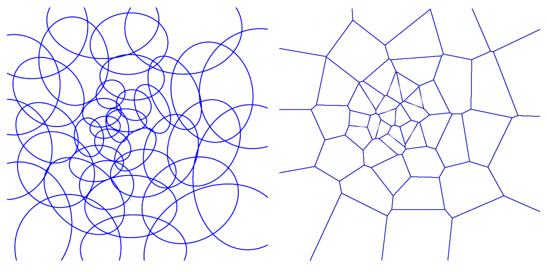

Fig. 1. Illustrative modeling of the acoustic space using a) EM-ML training and Gaussian mixtures, and b) K-means algorithm.

2.2. Algorithms proposed

In this section we present three approaches that take advantage of the power offered by K-means clustering, either applying it by itself or in combination with the standard GMM-posteriorgrams method in order to improve its results.

2.2.1. *Hierarchical K-means features*

Given that the hierarchical K-means by itself follows both desired characteristics of a good acoustic feature for pattern-matching applications we decided to obtain the desired feature vectors directly from the resulting clusters. To do so, we gather all frames assigned to each cluster in the final iteration and train a single Gaussian with them. A final GMM-like model is created by pooling all these Gaussians together and estimating their weights using the percentage of frames assigned to each cluster. Note that not a single EM-ML iteration is executed, which makes the system much faster to train than a standard GMM model.

2.2.2. *Discriminant-GMM features*

Discriminant-GMM modeling (DGMM) is composed of an initial standard EM-ML training to obtain a GMM model followed by a K-means training. In order to obtain the initial Gaussians for the GMM we have experimented with two possible methods: hierarchical K-means (as explained above) and Gaussian Splitting, which applies the same iterative approach as hierarchical K-means, but reassigs the frames using expectation maximization (EM) steps.

Once the GMM is obtained we apply K-means training to make features more discriminant. The initial K-means clusters are initialized using the Gaussian means. Next we assign each frame in the training data to its closest cluster (using euclidean distance) and we recompute the cluster means. As an alternative to this implementation (which considers isotropic clusters), we have also tested the possibility of performing a hard assignment of frames to the closest Gaussian according to their likelihood, and then recomputing all Gaussian statistics using only the data assigned to each Gaussian. In both cases we iterate the assignment-update stages until less that 1% of frames change cluster between any two consecutive iterations. Once the process converges, we obtain our DGMM by training the Gaussians only with the assigned frames like in the previous algorithm.

Given that this method builds upon standard GMM training, it is therefore useful both to train new models and to adapt existing

GMM models with additional data.

2.2.3. *BinaryGrams features*

A disadvantage of any posteriorgram-based technique is that its dimensionality is usually bigger than that of the original acoustic features (e.g. MFCC) obtained from the data. In order to solve this we rely on techniques derived from research on speaker ID [8] to obtain binary features from the data. In particular, given a feature vector obtained from any of the previous two models, we set to 1 the $\alpha\%$ of values with highest posterior probability, and to 0 the rest. By selecting the Gaussians closest to a given input feature vector we are indicating which regions in space it is closest to. In practical terms, given that each dimension of the resulting feature vector is now represented by a single bit, we are reducing the storage requirements by *sizeof(float)* (which depends on the computer architecture being used). In our implementation we use $\alpha = 10\%$.

3. QUERY-BY-EXAMPLE SYSTEM

Query-by-example search is a possible application for the features proposed. Given an audio query composed of one or multiple spoken words, we strive at finding the location(s) where it is spoken within some reference database. Speech recognition-based systems either transcribe the reference and query or obtain their phoneme latices in order to compare their values in search for matches. As discussed earlier, these methods require a priori knowledge of the language we want to process and transcribed and aligned data for training. By using pattern-matching techniques we can compare directly the signals at the acoustic level.

The pattern-matching approach we use is defined as follows. Given two sequences, $X = \{x[1], \dots x[i], \dots, x[N_x]\}$ and $Y = \{y[1], \dots y[j], \dots, y[N_y]\}$ of acoustically-derived features, respectively obtained from the query and the reference, we compare them using a DTW-like algorithm. The standard DTW algorithm returns the optimum alignment between any two sequences by finding the optimum path between their beginning $(i, j) = (1, 1)$ and end $(i, j) = (N_x, N_y)$ points. In our case we constraint the query signal to match between start and end, but we allow the reference signal to start its alignment at any position $(1, \cdot)$ and finish whenever the dynamic programming algorithm reaches $i = N_x$. Although we do not set any global constraints, the local constraints are set so that a maximum 2-times or $\frac{1}{2}$-times warping is allowed by choosing the path that minimizes the cost to reach position (i, j) as

$$\text{cost}(i,j) = \\ d(i,j) + \min \begin{cases} D(i-2,j-1))/(\#(i-2,j-1)+3) \\ D(i-2,j-2))/(\#(i-2,j-2)+4) \\ D(i-1,j-2))/(\#(i-1,j-2)+3) \end{cases} \quad (1)$$

Where $D(i, j)$ is the accumulated (non-normalized) distance of all optimum paths until position (i, j), $d(i, j)$ is the local distance between frames x_i and y_j from both compared sequences, and $\#(i, j)$ is the number of jumps of the optimum path until position (i, j). Note than when normalizing the different possible paths we slightly favor the diagonal match. The local distance $d(i, j)$ is defined in Eq. 3 or 2 depending on whether the features are binary or real-valued, respectively, where K is the dimension of the feature vectors.

$$d(i,j) = -\log\left(\frac{\sum_{k=1}^{K}(x_k[i] \wedge y_k[j])}{\sum_{k=1}^{K}(x_k[i] \vee y_k[j])}\right) \quad (2)$$

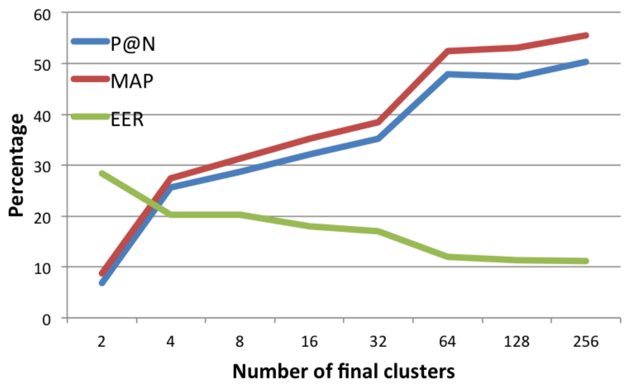

Fig. 2. P@N, MAP and EER for the Hierarchical K-means approach using different numbers of clusters

$$d(i,j) = \frac{< x[i], y[j] >}{||x[i]|| \cdot ||y[j]||} \quad (3)$$

4. EXPERIMENTAL SECTION

We tested the feature extraction proposed in a query-by-example task by using a 4h subset of the ESTER corpus [9]. This corpus is the same that is used in [10] and contains 4 different French broadcast news shows, recorded on the same day, from different radio stations. The recordings have been split into 2915 silence bounded segments of varying length, with an average of 4.87 seconds each. Then, 20 keywords have been extracted. These appear between 13 and 284 times in the reference, the query keyword having been eliminated from the test. Each keyword is spoken in average by 22.55 speakers. The objective of this test is to correctly find the sequences in which each of these queries appear, regardless of where they appear within the segment.

4.1. Evaluation criteria

For each query-reference segment pair the system computes a distance by using either equation 2 or 3, depending on the features being either real-valued or binary. For the case of MFCC features we apply an offset to the term inside the $log()$ in equation 2 in order to avoid negative values inside the $\log()$. Once all distances for a query have been computed, P@10, MAP and EER performance metrics are obtained. P@N indicates the percentage of correct matches that appear ranked in the top N results, where N is the number of documents in which the query appears in the reference database. MAP is the mean average precision, i.e. the mean of precision scores after each keyword occurrence has been retrieved. Finally, EER is the percentage error where false acceptances and false rejections are equal. In all cases except for the EER, the higher the percentage, the better. Such performance metrics are obtained for each query independently and then averaged over all queries.

4.2. Evaluation Results

The baseline acoustic features used throughout the experiments are 39-dimensional MFCC features (12 Cepstra + log energy + their deltas and double deltas) as in [10].

First we perform an analysis of the correct dimensionality of the feature vector (i.e. the number of Gaussians we need to obtain from the training data). Figure 2 shows all three performance metrics computed for the Hierarchical K-means system computed for a dimensionality between 2 and 256. The bigger the dimensionality, the better the system performs. Therefore, the choice of the right dimensionality will depend on the computation and feature storage limitations. We observe that after 64 Gaussians results show a slower increase. We therefore choose 64 as the dimension for our system.

Next, Table 1 compares the performance for all implementation combinations we described in section 2. It first shows results for a system using just the MFCC-39 features and the baseline GMM posteriorgrams system results. Then we show the results obtained with the three different systems proposed: using Hierarchical K-means alone, using the discriminant-GMM approach and the binarization of the previous two. We observe that MFCC-39 features are the worse performing in general, which was expectable as such features are rather speaker dependent. The baseline GMM posteriorgram approach obtains much improved performances than MFCC-39 both when initializing the model using hierarchical K-means clustering and Gaussian splitting (GSplit).

The Hierarchical K-means model was tested using two alternative implementations: one which trains the final Gaussians by assigning frames to the Gaussian whose likelihood is greater (indicated as "Lkld dist"), another (tested for completeness) simply uses the euclidean distance between a frame and each of the clusters as the feature vector. As expected, results using the posteriorgrams from the derived Gaussians perform much better than simple euclidean features and the baseline, probably as they take into account the Gaussian weights and variances.

For the discriminant-GMM approach we show performances for all combinations of K-means and Gaussian splitting initialization, and K-means and Likelihood-based frames reassignment post EM-ML. In our experiments the optimum values for all metrics are obtained when we do an initialization through Gaussian Splitting and Likelihood-based reassignment. These represent an increase in P@N of 3.7% and MAP of 1.3% absolute over the best GMM-posteriorgram approach tested, as well as a reduction in EER of 1.1% absolute.

Finally, for all the systems proposed we show results when we binarize the features turning the best $\alpha = 10\%$ values in each frame to 1. In general all results worsen and we get performances similar to those of MFCC-39 features, although with much smaller footprints. For example, for architechtures where a float occupies 32 bits, we would spend 1248 bits to store an MFCC-39 feature, 2048 bits for a GMM-based posteriorgram feature and only 64 bits for a binarygram.

5. CONCLUSIONS

Recent developments in acoustic pattern-matching allow the processing of audio data without the need for large transcribed datasets, standard in most HMM-based speech processing algorithms. A very important step towards the success of these methods relies on the front-end module, which needs to obtain features derived from the audio data that are content-aware and independent of speaker variability and changes in the acoustic conditions. In this paper we proposed two improvements to the commonly used GMM-posteriorgram front-end by combining it with the K-means algorithm in order to increase its acoustic class discriminability while maintaining its speaker independence. Experiments show that the methods proposed improve over baseline techniques. In addition, we also propose a binarized representation of the feature vectors, useful where storage capacity are limited.

Table 1. Performance results

Description	P@N	MAP	EER
MFCC-39 feats	41.6 %	45.2%	16.2 %
Standard GMM Posteriorgrams			
K-means+EMML	41.4%	50.9%	12.3%
GSplit+EMML	45.7%	52.3%	12.2%
Hierarchical K-means			
Lkld dist.	47.8%	45.4%	12%
Euclidean dist.	38%	40.9%	16.8%
Discriminant-GMM			
K-means+EMML+K-means	47.3%	52.4%	11.8%
K-means+EMML+Lkld-based	46.3%	52.7%	12.1%
GSplit+EMML+K-means	46.5%	51.7%	11.8%
GSplit+EMML+Lkld	**49.4%**	**53.6%**	**11.1%**
BinaryGrams			
K-means	41.6%	45.9%	15.7%
K-means+EMML+K-means	41.7%	45.8%	15.7%
K-means+EMML+Lkld	40.6%	43.5%	**15.2%**
GSplit+EMML+K-means	**42.7%**	**46.1%**	15.6%
GSplit+EMML+Lkld-based	41.3%	44.5%	15.3%

6. ACKNOWLEDGEMENTS

We would like to thank Guillaume Gravier at IRISA-Rennes for sharing with us the databases and evaluation scripts we used in this work.

7. REFERENCES

[1] C. S. Myers and L. R. Rabiner, "A level building dynamic time warping algorithm for connected word recognition," *IEEE ASSP*, vol. 2, pp. 284–297, 1981.

[2] A. Park and J. Glass, "Unsupervised pattern discovery in speech," *IEEE-TASLP*, vol. 16, pp. 187–197, 2008.

[3] X. Anguera, R. Macrae, and N. Oliver, "Partial sequence matching using an unvounded dynamic time warping algorithm," in *PRoc. ICASSP*, 2010.

[4] A. Muscariello, G. Gravier, and F. Bimbot, "Audio keyword extraction by unsupervised word discovery," in *Proc. Interspeech*, 2009.

[5] A. Jansen, K. Church, and H. Hermansky, "Towards spoken term discovery at scale with zero resources," in *Proc. Interspeech*, 2010.

[6] G. Aradilla, J. Vepa, and H. Bourlard, "Using posterior-based features in template matching for speech recognition," in *Proc. ICSLP*, 2006.

[7] Y. Zhang and J. Glass, "Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams," in *Proc. ASRU*, Merano, Italy, December 2009, pp. 398–403.

[8] J-F Bonastre, X. Anguera, G. H. Sierra, and P-M Bousquet, "Speaker modeling using local binary decisions," in *Proc. Interspeech*, 2011.

[9] S. Galliano et al., "The ESTER evaluation campaign for the rich transcription of French broadcast news," in *Proc. Interspeech*, 2005.

[10] A. Muscariello, G. Gravier, and F. Bimbot, "Zero-resource audio-only spoken term detection based on a combination of template matching techniques," in *Proc. Interspeech*, 2011.