A ROBUST SPEAKER CLUSTERING ALGORITHM

J. Ajmera[*]

IDIAP
P.O. Box 592
CH-1920 Martigny, Switzerland
jitendra@idiap.ch

C. Wooters[†]

ICSI
1947 Center St., Suite 600
Berkeley, CA 94704, USA
wooters@icsi.berkeley.edu

ABSTRACT

In this paper, we present a novel speaker segmentation and clustering algorithm. The algorithm automatically performs both speaker segmentation and clustering without any prior knowledge of the identities or the number of speakers. Advantages of this algorithm over other approaches are: no need for training/development data, no threshold adjustment requirements, and robustness to different data conditions. This paper also reports the performance of the algorithm on different datasets released by NIST with different initial conditions and parameter settings. The consistently low speaker diarization error rate clearly indicates the robustness of the algorithm.

1. INTRODUCTION

The goal of a speaker segmentation system is to divide a speech signal into a sequence of speaker-homogeneous regions. Thus, the output of such a system provides the answer to the question, “Who spoke when?” The output from such a system can be extremely useful for improving the quality of speech-to-text (STT) systems. Knowing when each speaker is speaking can be useful as a pre-processing step in STT systems. Such pre-processing may include vocal tract length normalization (VTLN), or speaker-adaptation. Automatic speaker segmentation is also useful in information retrieval and in the indexing of audio archives.

One of the difficulties with automatic speaker segmentation is deducing the appropriate number of clusters. Ideally, one would like to have as many clusters as there are speakers in the audio, which is not known *a priori*. It may also be desirable to have separate clusters for non-speech/music and silence.

Several clustering schemes have been proposed in the literature, most of which first segment and then cluster the data. The segmentation is either assumed to be known [1, 2, 3] or performed automatically prior to clustering [4, 5]. These approaches have limitations: in the former case, the correct segmentation is rarely known *a priori* for practical applications, and in the latter case, the errors made in the segmentation step are not only difficult to correct later, but can degrade the performance of the subsequent clustering step. In the proposed technique, we perform clustering directly on the data, deriving the segmentation (according to the clusters) in the process.

The proposed algorithm for speaker segmentation deduces the number of clusters automatically while optimizing a likelihood-based objective function. The algorithm runs iteratively, where the likelihood of the data along the best segmentation path (Viterbi score) increases until it reaches an optimal point and then it begins decreasing. We stop the process at the maximum value in the likelihood function. An important property of this algorithm is that it does not have a threshold term to adjust, for which a development test set is generally required. The algorithm is quite robust to different initial conditions and choice of acoustic feature vectors. These properties are demonstrated with the help of a number of experiments.

2. SPEAKER CLUSTERING ALGORITHM

As shown in Fig. 1, our speaker clustering algorithm is based on an ergodic hidden Markov model (HMM) formalism where the number of states in the HMM is equal to this initial number of clusters. Each state is composed of a set of S sub-states. These sub-states impose a minimum duration on the model. Thus, each state of the HMM is a cluster and is expected to represent a single speaker. The probability density function (PDF) of each state is assumed to be a Gaussian mixture model (GMM) with M Gaussian components, which are shared among all sub-states.

In the absence of any *a priori* information about the

[*] This work was supported by the Swiss National Science Foundation through project no. 2100-65067.01 on ”AudioSkim”

[†] This material is based upon work supported by the Defense Advanced Research Projects Agency Information Awareness Office EARS program. ARPA Order No. N614, Program Code No. 2E20, issued by DARPA/CMO under Contract No. MDA972-02-C-0038 as part of a subcontract to ICSI by SRI International.

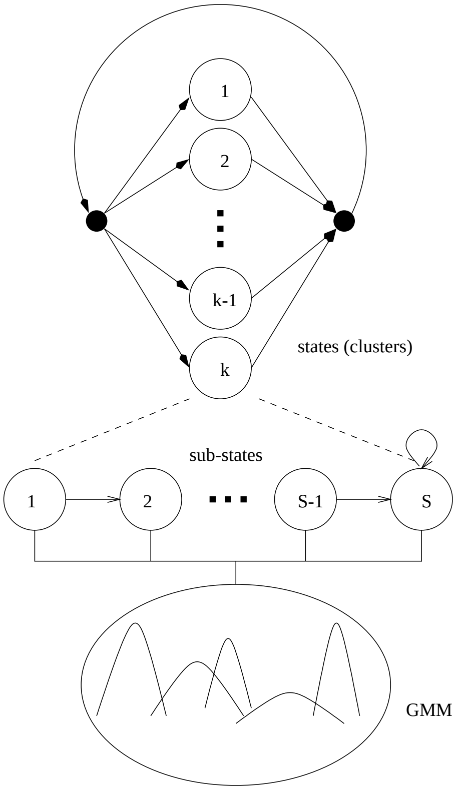

Fig. 1. HMM Topology Used for Clustering

number of speakers (and hence number of clusters), we start by over-clustering the data. The term “over-clustering” refers to the process of segmenting the data into an initial number of clusters K, where K is greater than the expected number of speakers in the audio file. A result of over-clustering is that data from a single speaker is likely to be assigned to different clusters. Thus, our goal is to identify clusters that have been assigned data from the same source and merge them. It will be shown later that the algorithm is not sensitive to the exact value of K as long as that value is large enough.

The initialization of the HMM parameters is done by assuming a uniform segmentation of the data in terms of K clusters and estimating the parameters of the cluster GMMs over these segments. K-means algorithm was also tried for this purpose, but our experiments showed that the algorithm is robust to this initialization mainly because of the large minimum duration that we impose.

The next step is to train this initial HMM topology. This is done by using the standard *Expectation-Maximization* (EM) algorithm. In the E-step, a segmentation of the data is obtained to maximize the likelihood of the data, given the parameters of the GMMs. This is followed by an M-step where the parameters of the GMMs are re-estimated/updated based on this segmentation.

The final step in the algorithm is cluster merging. A consequence of over-clustering is that data from a single speaker may be assigned to different clusters. Thus, a requirement of our algorithm is to identify clusters containing data from the same source and merge them. The details of cluster merging are explained in Section 2.1 below.

Once we have merged a pair of clusters, we return to the segmentation and training step. We repeat the process of segmentation-training-merging until there are no more clusters to be merged.

2.1. Cluster Merging

We begin with a high-level description of the overall problem: If $X = \{x_1, x_2, ..., x_N\}$[1] is the audio data to be segmented, we want to find the optimal number of clusters (k^*) and their acoustic models (Λ_k^*) that produce the “best” segmentation of the data (X) according to:

$$\Lambda_k^*, k^* = \arg\max_{\Lambda_k, k} p(X, q_{best}|\Lambda_k, k) \qquad (1)$$

where q_{best} is the Viterbi segmentation path with the highest likelihood. Thus, we want to find the set of clusters and their acoustic models that maximize the likelihood of the data and the associated segmentation based on this HMM topology.

Since we do not want to consider all possible values for k, we begin by choosing a maximum value ($k = K$). Then, through the process of cluster merging, we reduce the value of k until we find an optimal number of clusters (k^*) and their acoustic models (Λ_k^*) according to (1). However, if we merge two clusters, resulting in one fewer state in the HMM topology, then the total number of parameters in the HMM will be reduced. Because the same amount of data will be modeled using fewer parameters, the likelihood of the data given the model will decrease monotonically with decreasing model size. Since the merging process will not result in a maximum in the likelihood function, we would need to choose a threshold at which to stop merging.

Ideally, we would like to find a method of selecting clusters for merging such that a correct merge (*i.e.* a merge involving clusters of data from the same speaker) will produce an increase in the objective function (1) and an incorrect merge will result in a decrease. A common method of selecting between competing models is to use the Bayesian Information Criterion (BIC) [1]. BIC imposes a trade-off between model quality and model complexity. Using BIC as a merging criterion, two clusters would become a candidate for merging if the following is true:

$$\log p(D|\theta) - \frac{1}{2}\lambda K \log N \geq \log p(D_a|\theta_a) + \log p(D_b|\theta_b) \qquad (2)$$

[1] Generally this is a sequence of acoustic feature vectors extracted from the audio waveform at a regular time interval *e.g.* every 10ms.

where:

- D_a and D_b represent the data in two clusters and θ_a and θ_b represent the parameters of the PDFs of these two clusters respectively.
- D is the data from $D_a \cup D_b$ and θ represents the parameters of PDF of D.
- λ is *ideally* set to 1.0
- N is the number of data points in $\{D\}$
- K is the difference in the number of parameters between θ_a and θ_b

BIC provides a simple way to decide when to stop merging. However, we found in our experiments that we have to tune the value of λ to get best results. Moreover, this value changes with the data conditions and so a development dataset is required to estimate the optimal value of this parameter. Thus, we would like to eliminate the use of threshold (λ) to avoid the need of finding an optimal value of this parameter.

2.1.1. New Merging Criterion

If we use BIC, but keep the number of parameters constant, we eliminate the need for the penalty term in (2). Thus, when we merge two clusters, we simply model the PDF of the new cluster using a model containing a number of parameters equal to the sum of the number of parameters of the two merged clusters. As with BIC, the objective function in (1) increases for correct merging (merging of two clusters having data from the same source) and decreases for incorrect merging. We define our merging criterion as follows:

- Let M_a and M_b represent the number of parameters (Gaussian components) in θ_a and θ_b respectively.
- Let us hypothesize a new cluster having data $D = D_a \cup D_b$ with a PDF modeled by a GMM parameterized by θ with $M_a + M_b$ number of Gaussian components.

Given these conditions, a pair of clusters (D_a and D_b) becomes a candidate for merging if the following is true:

$$\log p(D|\theta) \geq \log p(D_a|\theta_a) + \log p(D_b|\theta_b) \qquad (3)$$

This is similar to BIC, except that the number of parameters in θ is equal to the sum of the number of parameters in θ_a and θ_b. By keeping the number of parameters constant from one iteration to the next, we have eliminated the need for the penalty term ($-\frac{1}{2}\lambda K \log N$). We have verified empirically that selecting candidates for merging using this criterion always results in an increase in the objective function associated with (1).

After every new segmentation-training step, we look for the best pair satisfying (3). In the case of many such candidate pairs, we choose the pair that maximizes the difference of the terms of left hand side and right hand side of (3). The merging is stopped when there are no suitable candidates satisfying (3).

Thus we now have a way to merge clusters without the use of any tunable parameters. Additionally, this method provides a fully automatic stopping criterion. However, there are a few "hyper-parameters" in this algorithm, namely the initial number of clusters (K), the initial number of Gaussian components in each cluster (M), the type of initialization used to create the clusters, and the set of acoustic features used to represent the signal. In Section 3 we present the results of several experiments in which we explore the effects of varying the hyper-parameters.

3. EXPERIMENTS AND RESULTS

3.1. Evaluation Criterion

Evaluation of the algorithm was done using NIST's RT-03S scoring script (`SpkrSegEval-v21.pl`). This calculates a time-based score (error) that is the percentage of speaker time not attributed correctly to a reference speaker. Thus, a score of 0.0 would represent a perfect segmentation. Also, it is possible to have an error > 100.0 because of the inclusion of false alarms. The error is calculated as:

$$Error = \frac{\sum_{s=1}^{S}\{dur(s)*(max(N_{ref}(s),N_{sys}(s))-N_{correct}(s))\}}{\sum_{s=1}^{S}\{dur(s)*N_{ref}(s)\}} \qquad (4)$$

where the speech data is divided into contiguous segments whose boundaries are defined by all speaker change points (including both reference and hypothesized speakers) and where, for each segment s:

$dur(s)$ = the duration of s
$N_{ref}(s)$ = the # of reference speakers speaking in s
$N_{sys}(s)$ = the # of system speakers speaking in s
$N_{correct}(seg)$ = the # of reference speakers speaking in s for whom their matching (mapped) system speakers are also speaking in s.

3.2. Data

The algorithm was tested on 3 different sets of data released by NIST, namely *dryrun*(data used for preliminary experiments), *devdata* (data used as development data) and *evaldata* (data used for final RT03s evaluation). *dryrun* consists of 6 10-minutes audio segments, and *devdata* and *evaldata* consist of 3 30-minutes audio segments each.

Note: not all the experiments were carried out on 3 datasets. We experimented with different hyper-parameters at different stages on different datasets. Also, the results are shown for the complete datasets to avoid presenting too many numbers.

3.3. Baseline System

We used default values shown in Table 1 for all the hyper-parameters in each of the experiments listed below, unless otherwise noted. The performance of the baseline system is shown in Table 2.

Initialization	Uniform
Initial number of Gaussians (M)	5

Initial number of clusters (K)	15(*dryrun*)
	40(*devdata*), 40 (*evaldata*)

Minimum duration (S)	2 seconds
Feature type	LPC Cepstrum (LPCC)
Feature vector frequency	100 Hz

Table 1. Default values for the “hyper-parameters”

Table 2 presents results of the baseline system for the three datasets.

Dataset	Error
dryrun	28.85%
devdata	26.11%
evaldata	21.40%

Table 2. Baseline results for the three datasets.

The results on *evaldata* were submitted as part of RT-03s (http://www.nist.gov/ speech/tests/rt/rt2003/spring) evaluation and the performance of the system was highly competitive compared to other submitted systems. However, as seen in table 1, there are a number of hyper-parameters, which can be seen of as ’tunable’ parameters. We verified with the help of a series of experiments that the algorithm is not highly sensitive to any of these parameters. This together with experiments on different datasets show the robustness of the algorithm. These experiments are summarized in next subsections.

3.4. Initialization

As mentioned earlier, two different initialization were tried on *dryrun* dataset. Table 3 presents results for this experiment:

We realized that because of the minimum duration constraints and an iterative EM algorithm, the initialization does not make a big difference. Thus, for the subsequent experiments, we tried only uniform initialization.

Initialization	Error
Uniform	28.85%
K-means	29.56%

Table 3. Results for two different initialization schemes on *dryrun* dataset

3.5. Experiments with different acoustic features

Table 4 presents the results of using different acoustic features in the segmentation. In addition to the default 12-LPCC features, we tried 19-Mel-frequency cepstral coefficients (MFCC).

$Dataset$	$FeatureType$	$Error$
dryrun	LPCC	28.85%
dryrun	MFCC	29.22%
devdata	LPCC	26.11%
devdata	MFCC	25.13%
evaldata	LPCC	21.40%
evaldata	MFCC	20.79%

Table 4. Results obtained with alternate features.

As expected, the performance of LPCC and MFCC features in all the cases are comparable. However, while analyzing performance on individual files of each dataset, it was noticed that MFCCs work better in case of noisy conditions, while LPCCs work better during clean speech.

3.6. Experiments with minimum duration

Table 5 presents the results obtained by varying the minimum duration of each cluster. These experiments were also carried out on *devdata*.

Minimum Duration (secs)	Error
2	26.11%
3	26.55%
4	26.18%

Table 5. Results obtained with different minimum durations

Results in table 5 show that the algorithm is not sensitive to the minimum duration that we impose. However, it also depends on the average duration of the speaker segments. If there are many short segments, a large minimum duration may hurt. Thus, if we have an *a priori* information about this, it can be used in the algorithm.

3.7. Experiments with the number of initial clusters K

We have verified experimentally that this algorithm is not overly sensitive to the initial number of clusters, as long as we start with a reasonably large number. Generally, we have observed that a number of clusters greater than the number of minutes of data is sufficient.

Tables 6 shows results for varying the number of initial clusters.

dataset	K	Error
dryrun	15	28.85%
dryrun	30	28.35%
devdata	30	29.28%
devdata	40	26.11%
devdata	50	25.80%

Table 6. Results obtained by varying the number of initial clusters.

Again, results in Table 6 show that the algorithm works well as long as we start from large enough number of clusters. A simple trick is to choose a number greater than the number of minutes of audio data. However, it should be noted that a large number of initial clusters results into a higher computational complexity.

3.8. Experiments with the number of initial Gaussians (M)

Note that we use a small number of Gaussians compared to numbers used in the speaker recognition framework. In the speaker recognition framework, the goal is to make a robust model for each speaker, and hence a large number of Gaussians are employed. However, the goal here is to make discriminative models for the speakers in a single audio stream. Thus, we need fewer Gaussian components to estimate the PDFs of each cluster. Generally, we choose five for this purpose, but we verified in our experiments that the performance of the algorithm does not appear to be overly sensitive to this choice. Table 7 presents the results of experiments in which we varied the number of Gaussian components in each initial cluster while experimenting on *devdata*.

M	Error
5	26.11%
10	27.44%

Table 7. Results obtained for different number of Gaussians (M) for *devdata*

From the results in Table 7, we see that the algorithm is not very sensitive to the number of Gaussians employed for each cluster in the beginning. Thus, for all our experiments (including those submitted to NIST), we used 5 Gaussians.

4. CONCLUSION

In this paper, we presented a speaker clustering algorithm and showed its robustness to different data conditions with the help of numerous experiments. The algorithm is basically an HMM based agglomerative clustering framework where the clusters are merged in successive iterations to finally reach the optimal number of clusters. A merging criterion is defined for this purpose which always results in an increase in a likelihood based objective function. The important property of this algorithm is that it does not rely on an adjustable threshold/parameter, which not only makes the algorithm robust but also eliminates the need for a development dataset.

5. REFERENCES

[1] S. S. Chen and P. S. Gopalakrishnan, “Speaker, environment and channel change detection and clustering via the Bayesian information criterion,” Tech. Rep., IBM T.J. Watson Research Center, 1998.

[2] M. Sugiyama, J. Murakami, and H. Watanabe, “Speech segmentation and clustering based on speaker features,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 395–398, 1993.

[3] A. Solomonoff, A. Mielke, M. Schmidt, and H. Gish, “Clustering speakers by their voices,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 757–760, 1998.

[4] M . A. Siegler, U. Jain, B. Raj, and R. M. Stern, “Automatic segmentation, classification and clustering of broadcast news audio,” *DARPA Speech Recognition Workshop, Chantilly*, pp. 97–99, Feb 1997.

[5] T Hain, S. E. Johnson, A. Turek, P. C. Woodland, and S. J. Young, “Segment generation and clustering in the HTK broadcast news transcription system,” *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pp. 133–137, 1998.