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. Our algorithm uses "standard" speech processing components and techniques such as HMMs, agglomerative clustering, and the Bayesian Information Criterion. However, we have combined and modi£ed these so as to produce an algorithm with the following advantages:

- No threshold adjustment requirements
- No need for training/development data
- Robustness to different data conditions

This paper also reports the performance of this algorithm on different datasets released by the U.S. National Institute of Standards and Technology (NIST) with different initial conditions and parameter settings. The consistently low speaker-diarization error rate clearly indicates the robustness and utility 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?" Knowing when each speaker is speaking is useful as a pre-processing step in speech-to-text (STT) systems to improve the quality of the output. Such pre-processing may include vocal tract length normalization (VTLN) and/or speaker adaptation. Automatic speaker segmentation may also be useful in information retrieval and as part of the indexing information of audio archives.

Dividing an audio recording into speaker-homogeneous regions presents many challenges. One challenge is to identify the locations of the boundaries between speakers — the "speaker segmentation" problem. Another challenge is to identify which portions of the recording belong to which speakers — the "speaker clustering" problem. Additonally, the speaker clustering problem requires that we correctly identify how many unique speakers occur in the recording.

Speech researchers have proposed many techniques for solving the "Who spoke when?" problem. Most of these methods £rst segment and then cluster the data. The segmentation is either assumed to be known *a priori* [1, 2, 3] or is performed automatically prior to clustering [4, 5]. However, approaches such as these, in which the segmentation and clustering are performed sequentially, have limitations. In the former case, the correct segmentation is rarely known *a priori* for practical applications. In the latter case, the errors made in the segmentation step can degrade the performance of the subsequent clustering step.

Sequential systems such as those mentioned above can be improved by iterating the segmentation and clustering steps. However, they still require some means of guessing the appropriate number of clusters, or a threshold to decide when the appropriate number of clusters has been attained. The approach we are proposing also iterates the segmentation and clustering steps. However, we have developed a fully automatic stopping criterion that produces the optimal number of clusters and the optimal segmentation without the use of thresholds.

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, then begins decreasing. (See Fig. 1.) 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 several experiments.

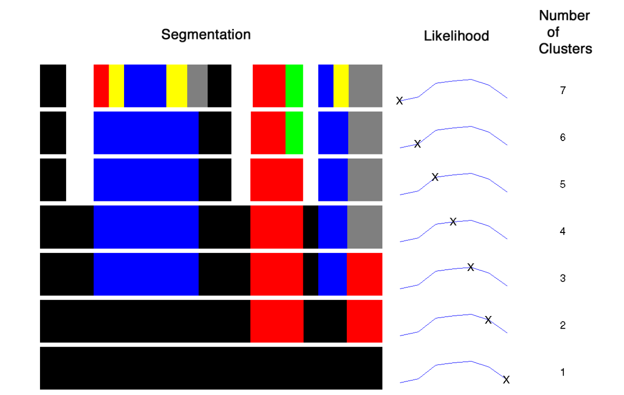

Fig. 1. Segmentation as a function of the number of clusters. The likelihood function has a maximum at the correct number of clusters, which in this case is three.

2. SPEAKER CLUSTERING ALGORITHM

As shown in Fig. 2, our algorithm is based on an ergodic hidden Markov model (HMM) formalism where the number of states in the HMM is equal to the 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 number of speakers (and hence number of clusters), we start by over-clustering the data. "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 £le. 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 relative to the actual number of speakers.

The £rst step in the algorithm is to initialize the parameters of the HMM. We perform this initialization using a uniform segmentation of the data in terms of K clusters and estimating the parameters of the cluster GMMs over these segments. We also tested the K-means algorithm for initialization. The results were not signi£cantly different, probably because of the large minimum duration that we impose in our cluster models.

The next step is to train the initial HMMs 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.

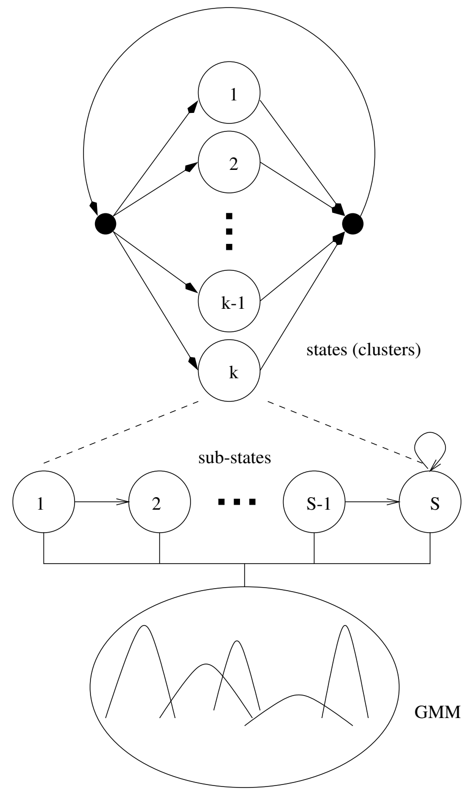

Fig. 2. HMM Topology Used for Clustering

The £nal 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, the algorithm requires a component that identi£es clusters containing data from the same speaker and merges 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 £nd the optimal number of clusters (k^*) and their acoustic models (Λ^*_k) that produce the "best" segmentation of the data (X) according to:

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

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

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, when two clusters are merged, the total number of parameters in the HMM decreases. Modeling the same amount of data using fewer parameters yields a lower likelihood score. Given that the merging process can only result in monotonically decreasing likelihoods, we will not observe a maximum in the likelihood function at any point other than the starting point. Therefore we need to choose a likelihood *threshold* to tell us when 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:

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

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 a "tweakable" parameter that 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 had to tune λ to get the best results. Moreover, the optimal value of λ changes depending on the data conditions and finding the optimal value requires the use of a development dataset that closely resembles the test dataset. In general, the algorithm would be more robust if thresholds such as λ could be eliminated.

2.1.1. New Merging Criterion

If we use BIC, but keep the number of parameters constant, we can eliminate the penalty term ($-\frac{1}{2}\lambda K \log N$) 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:

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

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 . We have verified empirically that selecting candidates for merging using this criterion does indeed result 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).

This method provides a fully automatic stopping criterion that does not require the use of any tunable parameters. 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:

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

where the speech data is divided into contiguous segments whose boundaries are de£ned 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 three different datasets released by NIST, namely *dryrun* data (data used for preliminary experiments), *devdata* (data used as development data) and *evaldata* (data used in the £nal RT-03S evaluation).[2] The *dryrun* data consists of six 10-minute excerpts from English broadcast news shows, while the *devdata* and *evaldata* each consisted of three half-hour English broadcast news audio segments. In the *evaldata* alone, there are 47 Male and 12 Female speakers.

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.

The results on *evaldata* were submitted[3] as part of the RT-03S[4] 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 as "tunable" parameters. We veri£ed 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, demonstrates the robustness of the algorithm. These experiments are summarized in next subsections.

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"

Dataset	Error
dryrun	28.85%
devdata	26.11%
evaldata	21.40%

Table 2. Baseline results for the three datasets.

3.4. Initialization

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

Initialization	Error
Uniform	28.85%
K-means	29.56%

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

We observed that the type of initialization does not have a signi£cant impact on the £nal results. We speculate that this is due to the minimum duration constraint and the use of an iterative EM algorithm. Thus, for the subsequent experiments, we used only uniform initialization.

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-

[2]Not all the experiments were carried out on all three datasets. We experimented with different hyper-parameters at different stages on different datasets. Also, all results are shown for complete datasets only (rather than individual £les) to avoid presenting too many numbers.

[3]The results submitted to NIST also involved another postprocessing stage of speech/non-speech discrimination. Since this module is not a focus of this paper, the results shown here are without any speech/non-speech discrimination, i.e. only based on speaker clustering.

[4]http://www.nist.gov/speech/tests/rt/rt2003/spring

LPCC features, we tried 19-Mel-frequency cepstral coef£-cients (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 the performance on individual £les of each dataset, we noticed that MFCCs worked better in noisy conditions, while LPCCs worked 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, the sensitivity also depends on the average duration of the speaker segments. If there are many short segments, a large minimum duration may result in a higher error rate. Thus, if we have any *a priori* information about average speaker durations, it can be used in the algorithm.

3.7. Experiments with the number of initial clusters

Table 6 presents results for varying the number of initial clusters (K). These results show that the performance of the algorithm improves as we increase the value of K. Since the algorithm is based on agglomerative clustering, K should be larger than the expected number of speakers. We have found that a good “rule of thumb” (for English broadcast news) is to choose K equal to or greater than the number of minutes of audio data. However, it should be noted that as the value of K increases, so does the computational complexity of the algorithm. This is due to the number of pairwise cluster comparisons that must be made during merging.

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.

3.8. Experiments with the number of initial Gaussians

Table 7 presents the results of experiments on the *devdata* in which we varied the number of Gaussian components in each initial cluster (M). Note that we use a small number of Gaussians relative to those used in the speaker recognition framework. In the speaker recognition framework, the goal is to make a robust model for each speaker, and hence thousands of Gaussians per speaker are employed. However, the goal here is to make discriminative models for the speakers in a single audio stream. Thus, we need far fewer Gaussian components to estimate the PDFs of each cluster.

M	Error
5	26.11%
10	27.44%

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

In the majority of our experiments (including those submitted to NIST), we used £ve gaussians per initial cluster ($M = 5$). We determined that the performance of the algorithm does not vary signi£cantly depending on the choice of M. In fact, the performance degrades slightly for larger values as can be seen in Table 7.

4. CONCLUSION

In this paper, we have presented a novel speaker clustering algorithm and demonstrated its robustness to different data conditions. The algorithm uses an iterative, agglomerative clustering technique based on an HMM framework. Pairs of clusters are merged in successive iterations and merging stops automatically when the optimal number of clusters has been determined. We de£ned a BIC-like merging criterion that we use to choose which clusters to merge and when to stop merging. This merging criterion produces an increase in a likelihood-based objective function for correct merges and a decrease for incorrect merges. An important property of our merging criterion is that it does not use an adjustable threshold. The absence of such “tweakable” parameters not only inproves the algorithm’s robustness but also eliminates the need for a development dataset.

Acknowledgements

This work was supported by project MULTI (Swiss NSF project no. 2000-068231.02/1), project M4 (EC-IST project no. 2001-34485) and the Swiss National Center of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM)2. The NCCR is managed by the Swiss NSF on behalf of the federal authorities. This Material is also based upon work supported by the Defense Advanced Research Projects Agency Information Awareness Of£ce 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.

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, classi£cation 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.