ONLINE SPEAKER CLUSTERING

Daben Liu, Francis Kubala

BBN Technologies
10 Moulton Street, Cambridge, MA 02138
dliu@bbn.com, fkubala@bbn.com

ABSTRACT

This paper describes a set of new algorithms that perform speaker clustering in an online fashion. Unlike typical clustering approaches, the proposed method does not require the presence of all the data before performing clustering. The clustering decision is made as soon as an audio segment is received. Being causal, this method enables low-latency incremental speaker adaptation in online speech-to-text systems. It also gives a speaker tracking and indexing system the ability to label speakers with cluster ID on the fly. We show that the new online speaker clustering method yields better performance compared to the traditional hierarchical speaker clustering. Evaluation metrics for speaker clustering are also discussed.

1. INTRODUCTION

The goal of speaker clustering is to identify all speech segments from the same speaker in an audio episode and assign them a unique label. The true number of speakers involved in the episode is usually unknown, making it a difficult problem. The clustering output is commonly used in speaker adaptation for speech recognition to collect sufficient data from the same speaker [2][3][4][5]. The clustering information itself is very useful in labeling and organizing speakers in speaker tracking or audio indexing applications [1].

Existing clustering algorithms, such as the widely used hierarchical clustering [2][3][4][5], typically require that all the data be present before clustering can be performed. In these algorithms, different numbers of clusters are hypothesized based on local similarity or distance measures, e.g. building a dendrogram. Then a global criterion is used to find the best number of clusters. There are two drawbacks to this approach. First, the process is non-causal. It cannot be used in an online time-critical system or a system that requires incremental cluster information. Secondly, trying different number of clusters is computationally expensive and time consuming. In dendrogram-based clustering, for example, the computational complexity increases exponentially with the number of audio segments.

In this paper, we propose an online speaker clustering algorithm that does not require the presence of all the data. The clustering decision can be made whenever a new audio segment is received. We start by a brief overview of a hierarchical speaker clustering algorithm [2]. Then we introduce new algorithms for online speaker clustering. We present further improvement to the algorithms by studying the behavior of the decision criterion and the distance measure. Finally, we present the evaluation metrics and comparative results.

The input to the online speaker clustering application is usually audio segments generated from automatic speaker segmentation, which is not in the scope of this paper. Interested readers are referred to [1][8] for more details. All the experiment results here are reported on reference segmentations.

2. HIERARCHICAL SPEAKER CLUSTERING

In 1996, we developed an automatic speaker clustering algorithm to improve the performance of unsupervised speaker adaptation [2]. There were other approach [3][4][5] using different distance measures and global criterion in determining number of clusters, all of which fell into the hierarchical speaker clustering framework.

Consider a collection of segments $S = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n\}$ with each $\mathbf{s}_i$ denoting an audio segment that is represented by a sequence of feature vectors. A hierarchical speaker clustering (HC) algorithm can be described as follows:

Algorithm 1.[1] (Hierarchical Clustering)

1. **begin initialize** $c \leftarrow n$
2. **do**
3. find the nearest pairs in the c clusters, say, $\mathbf{s}_i$ and $\mathbf{s}_j$
4. merge $\mathbf{s}_i$ and $\mathbf{s}_j$
5. calculate the global criterion, save to $G(c)$
6. $c \leftarrow c - 1$
7. **until** $c = 1$
8. **return** $c \leftarrow \arg\min_c G(c)$ and the c clusters
9. **end**

where c is the hypothesized number of clusters and $G(c)$ is the global criterion to be minimized. There are two strengths in hierarchical method. Not only is it able to find the closest pairs by comparing distances among all the available segments, it can also compare different clustering strategies globally to pick the best. Neither of these advantages is available in the online case.

To calculate the distance, we used the generalized likelihood ratio (GLR). A Gaussian distribution, $N(\mu_i, \Sigma_i)$, is estimated from the sequence of feature vectors of each audio segment $\mathbf{s}_i$. The GLR between $\mathbf{s}_i, \mathbf{s}_j$ can be expressed as follows:

$$GLR(\mathbf{s}_i, \mathbf{s}_j) = \frac{L(\mathbf{s}_c; \mu_c, \Sigma_c)}{L(\mathbf{s}_i; \mu_i, \Sigma_i) L(\mathbf{s}_j; \mu_j, \Sigma_j)} \quad (1)$$

where $\mathbf{s}_c$ is the union of $\mathbf{s}_i$ and $\mathbf{s}_j$, and L(.) is the likelihood of the data given the model. We used within-cluster dispersion

[1] The representation style for algorithms is adopted from Duda, et al. [7]

penalized by number of clusters for global clustering selection, which is expressed as follows:

$$G(c) = |\sum_{j=1}^{c} N_j \Sigma_j| \sqrt{c} \qquad (2)$$

where c is the number of clusters, N_j is the number of feature vectors in cluster j, and Σ_j is the covariance matrix of cluster j. $|.|$ denotes the determinant.

The HC algorithm worked well for unsupervised speaker adaptation. It improves the word-error-rate as much as the hand-labeled ideal clustering [2]. In this paper, we will examine the usefulness of the same GLR distance and dispersion criterion for online speaker clustering.

3. ONLINE SPEAKER CLUSTERING

3.1. Leader-follower clustering

Duda, et al, present a generic approach called leader-follower clustering (LFC)[7], for *k*-means type clustering. The basic idea is to either alter only the cluster centroid most similar to a new pattern being presented so that the cluster is somewhat like the new pattern, or make a new cluster with the new pattern if none of the existing clusters are similar. The generic algorithm can be described as follows:

Algorithm 2 (LFC)

1. **begin initialize** η, θ
2. $\mathbf{w}_1 \leftarrow \mathbf{s}$
3. **do** accept new **s**
4. $j \leftarrow \arg\min_{j'} \| \mathbf{s} - \mathbf{w}_{j'} \|$
5. **if** $\| \mathbf{s} - \mathbf{w}_j \| < \theta$
6. **then** $\mathbf{w}_j \leftarrow \mathbf{w}_j + \eta \mathbf{s}$ (update $\mathbf{w}_j$)
7. **else** add new $\mathbf{w} \leftarrow \mathbf{s}$ (create new cluster)
8. $\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\|$
9. **until** no more patterns
10. **return** $\mathbf{w}_1, \mathbf{w}_2, \ldots$
11. **end**

where **w** is the collection of clusters, **s** is the input pattern, θ is the threshold, and η is the learning rate.

This generic algorithm can be easily adapted for online speaker clustering. Considering **s** as the audio segments, substituting the $\|.\|$ operation with the GLR measures and updating $\mathbf{w}_j$ by re-estimating a new Gaussian distribution, we can directly use the LFC algorithm to perform online speaker clustering. The threshold θ is empirically estimated from held-out data. The normalization step 8 is dropped as it is irrelevant in our case.

3.2. Dispersion-based speaker clustering

Data-dependent thresholds are not desirable because they reduce the robustness of a system. Since we have a proven criterion for model selection – the within-cluster dispersion *G(c)*, we can implement an online speaker clustering algorithm without the need for a threshold. The proposed algorithm, which we call dispersion-based speaker clustering (DSC) is as follows:

Algorithm 3 (DSC)

1. **begin**
2. $\mathbf{w}_1 \leftarrow \mathbf{s}$
3. **do** accept new **s**
4. $j \leftarrow \arg\min_{j'} GLR(\mathbf{s}, \mathbf{w}_{j'})$
5. G1 $\leftarrow$ $G(\mathbf{w}, \mathbf{w}_j\{\mathbf{s}\})$ "**s** is merged with $\mathbf{w}_j$"
6. G2 $\leftarrow$ $G(\mathbf{w}, \mathbf{s})$ "**s** is made a new cluster"
7. **if** G1 < G2
8. **then** update $\mathbf{w}_j$ with **s**
9. **else** add new $\mathbf{w} \leftarrow \mathbf{s}$ (create new cluster)
10. **until** no more patterns
11. **return** $\mathbf{w}_1, \mathbf{w}_2, \ldots$
12. **end**

From experiments we observed that DSC had a tendency to underestimate the number of clusters. To see how effective the dispersion criterion would be, we conducted an oracle experiment where step 7 of Algorithm 3 was substituted by known conditions from a reference set. In other words, when the input **s** truly belong to an existing cluster, merge **s** to the cluster. Otherwise create a new cluster with **s**. In every iteration, we still computed the G1, G2. In Figure 1, we plot the difference, (G2-G1). Circle (∘) represents a merge situation in the reference, while Plus (+) denotes new clusters. The horizontal line is drawn at 0. Thus, if we were to use (G2-G1) as our criterion, any points below this zero line would incur a creation of new cluster and any points above would incur a merge. We can see that using dispersion alone is not effective in splitting the two different events.

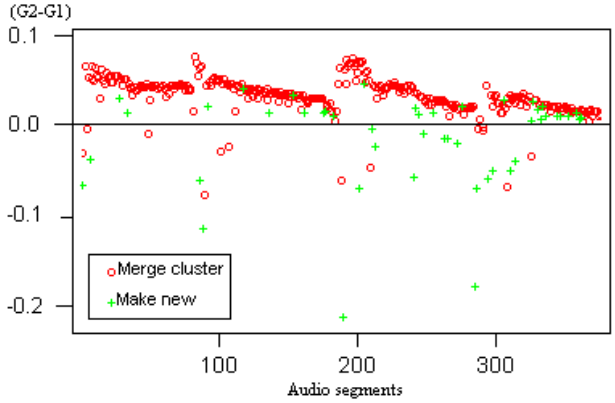

Figure 1. Dispersion difference plot for oracle experiment (The audio segments are indexed in the incoming order. Four audio files were processed sequentially, which is visible in the figure as the four descending traces. The descending is due to the penalty on number of clusters. With the increase of number of hypothesized clusters, the difference (G2-G1) decreases.)

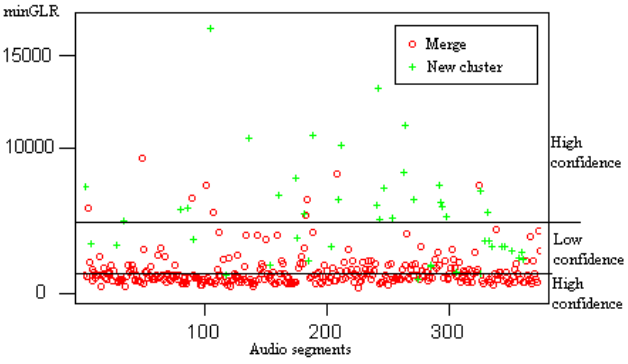

Figure 2. Minimum GLR plot for oracle experiment

From the oracle experiment, we observed that when some of the minimum GLR distances were very high, it was almost certain that new clusters should be created and vice versa for merging. We plot the minimum GLR for every incoming audio segment in Figure 2, again with the merging and creating events labeled by circles and plus. It is clear that one threshold, as in the LFC case, is not very successful in dividing the two different events. However, decisions could be made with high confidence at the upper and lower regions as indicated in the figure.

3.3. Hybrid speaker clustering

The above observation inspired a hybrid algorithm (HSC) that utilized both the dispersion and the GLR threshold. The algorithm is presented as follows:

Algorithm 4 (HSC)

1. **begin initialize** θ_U, θ_L
2. $\mathbf{w}_1 \leftarrow \mathbf{s}$
3. **do** accept new **s**
4. $j \leftarrow \arg\min_{j'} GLR(\mathbf{s}, \mathbf{w}_{j'})$
5. $D_j \leftarrow \min GLR(\mathbf{s}, \mathbf{w}_j)$
6. G1 $\leftarrow$ $G(\mathbf{w}, \mathbf{w}_j\{\mathbf{s}\})$ "**s** is merged with $\mathbf{w}_j$"
7. G2 $\leftarrow$ $G(\mathbf{w}, \mathbf{s})$ "**s** is made a new cluster"
8. **if** $D_j < \theta_L$
9. **then** update $\mathbf{w}_j$ with **s**
10. **else if** $D_j > \theta_U$
11. **then** add new $\mathbf{w} \leftarrow \mathbf{s}$ (create new cluster)
13. **else if** G1 < G2
14. **then** update $\mathbf{w}_j$ with **s**
12. **else** add new $\mathbf{w} \leftarrow \mathbf{s}$ (create new cluster)
13. **until** no more patterns
14. **return** $\mathbf{w}_1, \mathbf{w}_2, \ldots$
15. **end**

where θ_U, θ_L are the upper and lower thresholds that define the high confidence regions. The re-introduction of thresholds are somewhat undesirable. However, since they only work at high confidence regions, they are less sensitive to data and thus more robust. The low confidence region in Figure 2 is still handled by dispersion analysis. Note that when $\theta_U = \theta_L$, HSC becomes LFC. When $\theta_U \to \infty \; and \; \theta_L \to -\infty$, HSC becomes the DSC algorithm.

4. EXPERIMENT RESULTS AND EVALUATION

We used the four half-hour broadcast-news episodes, named as file1, fil2, file3 and file4, of the NIST Hub4 1996 evaluation set as our development data to optimize the thresholds. The evaluation data was taken from the Hub4 1998 evaluation set, which consists of two 1.5-hour broadcast news episodes.

4.1. Evaluation metrics

We selected three commonly used methods – the misclassification rate, cluster purity, and Rand Index [6], which are described as follows:

Consider N speakers that are clustered into C groups, where n_{ij} is the number of segments from speaker j that are labeled with cluster i. Assume that n_{ij} has f_{ij} speech frames.

Given a one-to-one speaker-to-cluster mapping, any segment from speaker j that is not mapped is considered an error, the summation of which is denoted as e_j. Then

$$\text{Misclassification rate} = \sum_{j=1}^{N} e_j \Big/ \sum_{i=1}^{C}\sum_{j=1}^{N} n_{ij} \qquad (3)$$

We define the final mapping as the one that minimizes the misclassification rate. Note that we calculate the error rate at the segment level, where segments with different lengths are treated equally. The insensitivity to length is desirable when speaker clusters are the end-products of the application. For example, in a speaker retrieval application, any incorrectly retrieved segment should be considered as equally undesirable regardless of the duration of the segment. Another advantage of this error measure is that it provides a clear map of error distribution that is useful for error analysis and debugging.

Cluster purity is calculated at the frame level. For each cluster i, calculate the pure frames f_i by adding up the speech frames of the majority speaker in cluster i.

$$\text{Cluster purity} = \sum_{i=1}^{C} f_i \Big/ \sum_{i=1}^{C}\sum_{j=1}^{N} f_{ij} \qquad (4)$$

Cluster purity is of great interest when speaker clustering is used for speaker adaptation, which concerns more about the total amount of correctly classified data.

Rand Index gives the probability that two randomly selected segments are from the same speaker but hypothesized in different clusters, or the two segments are in the same cluster but from different speakers. Let $n_{i\bullet}$ be the number of segments in cluster i, and $n_{\bullet j}$ be the number of segments from speaker j,

$$\text{Rand Index} = \left[\frac{1}{2}\left(\sum_{i=1}^{C} n_{i\bullet}^{\,2} + \sum_{j=1}^{N} n_{\bullet j}^{\,2}\right) - \sum_{i=1}^{C}\sum_{j=1}^{N} n_{ij}^{\,2}\right] \Big/ \binom{n}{2} \qquad (5)$$

Rand Index is a theoretical measure that has been widely used for comparing partitions [6]. The lower the index, the higher the agreement is between two partitions. However, it does not provide any information on how the partitions are distributed and how the two partitions are related.

4.2. Experiment results

The thresholds for LFC and HSC were tuned on Hub4 1996 test set by minimizing the overall misclassification errors. Hub4 1998 data set provides the fair results. All the results were compared to the hierarchical clustering (HC) performance. In Table 1, we compare the hypothesized number of clusters from each algorithm to the true number of speakers in each episode.

Episode	true # speakers	# clusters			
		HC	LFC	DSC	HSC
file1	7	8	9	10	10
file2	13	15	17	13	17
file3	15	16	18	15	18
file4	20	16	22	17	22
eval98-1	79	45	69	39	67
eval98-2	89	90	92	58	91

Table 1. Comparison of number of speakers and clusters

Episodes	# segments	Misclassification rate (%)				Cluster Purity (%)				Rand Index (%)			
		HC	LFC	DSC	HSC	HC	LFC	DSC	HSC	HC	LFC	DSC	HSC
file1	81	5	2	6	6	99.8	100.0	98.6	99.9	2.6	0.1	3.1	1.1
file2	106	21	11	14	10	96.3	98.1	90.9	98.2	4.5	0.9	1.9	0.8
file3	101	11	13	26	10	99.1	97.3	83.3	99.7	3.1	3.6	5.9	2.9
file4	92	30	26	39	23	92.4	95.0	76.0	95.6	5.3	5.2	6.4	4.5
eval98-1	399	24	27	31	28	89.7	94.4	78.6	93.1	1.5	1.4	1.6	1.2
eval98-2	428	39	32	40	29	84.7	89.2	77.1	89.7	0.9	0.8	1.1	0.7

Table 2. Speaker clustering error analysis

All algorithms are able to hypothesize reasonable number of clusters on the Hub4 1996 set. On the fair test, LFC and HSC perform better than HC and DSC. DSC significantly underestimated number of clusters on both Hub4 1998 episodes.

Table 2 shows the clustering performance of different algorithms using the three evaluation metrics described in 4.1. In general, DSC performed the worst in all measures, suggesting that within-cluster dispersion measure alone might not be a good choice for online speaker clustering. Both LFC and HSC yielded comparable or better performance compared to the baseline HC. Even though LFC performed superior on file1, the superiority did not hold up on other episodes. HSC, however, performed consistently well on all data sets suggesting that HSC is more robust than LFC.

4.3. Run-time efficiency

The computational complexity for hierarchical speaker clustering increases exponentially with number of audio segments, no matter how many speakers there are in the data. However, the complexity for proposed online speaker clustering algorithms is linear to the number of input audio segments. We ran HC and HSC on a series of test sets that contain from 80 to 1000 segments. The elapsed-time difference is shown in Figure 3. The elapsed-time difference between online and offline clustering starts to become significant at around 200 segments.

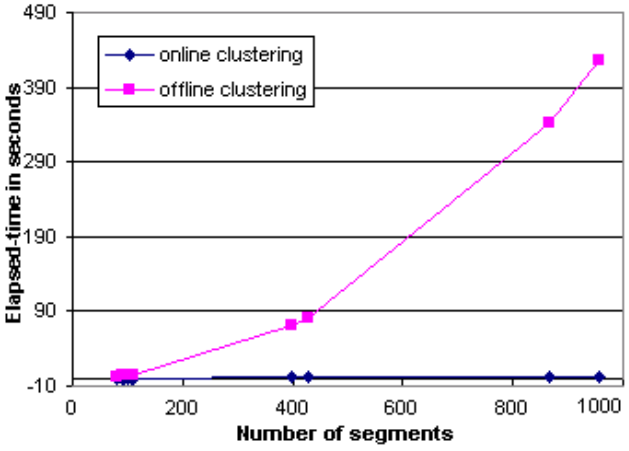

Figure 3. Speaker clustering elapsed-time vs. number of segments

5. CONCLUSION AND FUTURE WORK

We have developed three algorithms for online speaker clustering. All algorithms have shown the ability to automatically find reasonable number of clusters. LFC and HSC have shown promising results compared to the existing offline speaker clustering, while running much more efficiently.

Even though the DSC algorithm did not perform well in our evaluation, the notion of not having a threshold is still very attractive. Future efforts should focus on finding a more appropriate criterion that works better in the online clustering framework.

Intuitively, offline speaker clustering should work better given that it has more information available than what online clustering has. It also has the opportunity to find the global optimum. The test results show that there should be more rooms for improvement with offline speaker clustering. A hybrid system could be considered to use the online speaker clustering as the foreground process to produce clusters on the fly, while the offline clustering works in the background to globally refine the clustering results when there is sufficient data.

6. REFERENCES

1. Kubala, F., Sean Colbath, Daben Liu, Amit Srivastava, John Makhoul, "Integrated Technologies for Indexing Spoken Language," *Communications of the ACM*, Vol. 43, No. 2, February 2000
2. Jin, H., F. Kubala, R. Schwartz, "Automatic Speaker Clustering," *Proceedings of the DARPA Speech Recognition Workshop*, pp. 108-111, February 1997
3. Siegler, M., et al, "Automatic Segmentation Classification and Clustering of Broadcast News Audio," *Proceedings of the DARPA Speech Recognition Workshop*, pp. 97-99, February 1997
4. Hain, T., et al, "Segment Generation and Clustering in the HTK Broadcast News Transcription System," *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne*, VA, February 1998
5. Chen, S., P. Gopalakrishnan, "Speaker, Environment, and Channel Change Detection and Clustering via the Bayesian Information Criterion," *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, February 1998
6. Hubert, L., "Comparing Partitions," *Journal of Classification*, Vol 2, pp 193-218, 1985
7. Duda, R., P. Hart, D. Stork, "Pattern Classification," *John Wiley & Sons, Inc.*, Second Edition, 2001
8. Liu, D., F. Kubala, "Fast Speaker Change Detection for Broadcast News Transcription and Indexing," *EUROSPEECH'99*, Budapest, Hungary, Volume 3, Page 1031-1034, September 5-9, 1999