LARGE-SCALE COMMUNITY DETECTION ON SPEAKER CONTENT GRAPHS

Stephen H. Shum[1,2], *William M. Campbell*[2], *Douglas A. Reynolds*[2]

[1]MIT CSAIL, Cambridge, MA [2]MIT Lincoln Laboratory, Lexington, MA

sshum@csail.mit.edu, {wcampbell,dar}@ll.mit.edu

ABSTRACT

We consider the use of community detection algorithms to perform speaker clustering on content graphs built from large audio corpora. We survey the application of agglomerative hierarchical clustering, modularity optimization methods, and spectral clustering as well as two random walk algorithms: Markov clustering and Infomap. Our results on graphs built from the NIST 2005+2006 and 2008+2010 Speaker Recognition Evaluations (SREs) provide insight into both the structure of the speakers present in the data and the intricacies of the clustering methods. In particular, we introduce an additional parameter to Infomap that improves its clustering performance on all graphs. Lastly, we also develop an automatic technique to purify the neighbors of each node by pruning away unnecessary edges.

Index Terms— speaker clustering, community detection, modularity optimization, spectral clustering, random walk algorithms

1. INTRODUCTION

The state-of-the-art in speaker recognition has relied on the effectiveness of converting a speech signal into a vector-based representation. Techniques such as GMM supervectors [1], Joint Factor Analysis [2], and i-vectors [3] have enabled the application of matrix-based methods for quick and accurate recognition via inner products. The resulting ability to map an entire corpus of speech utterances to a set of vectors leads to questions about the structure of the underlying manifold. One way to analyze this space is to cover the manifold with a graph to use for analysis and recognition. In previous work by Karam [4, 5], graph embedding is proposed as both a way to visualize large data sets and as a starting point for speaker comparisons. Furthermore, the problem of query-by-example (QBE) using speaker content graphs was explored in [6].

In this paper, we build on the work in [4, 5, 6] and draw from the literature on community detection algorithms to perform large-scale clustering on speaker content graphs. That is, after summarizing a large collection of audio data using a sparse graph built according to the methods described in [6], we investigate tractable approaches to obtain insight into the graph structure, including how many speakers there are and which utterances correspond to whom.

An initial consideration of the speaker clustering problem on large datasets was presented in [7]. The related problem of large-scale speaker diarization was also explored for long recordings and small collections in [8]. Both works solely consider the use of agglomerative hierarchical clustering; neither consider the community detection algorithms or the sparse graph structure we investigate in this paper. Another similar work discovers the topics of YouTube videos [9], where the issue of scalability is resolved with a parallelized local partitioning technique before refining and merging clusters in a post-processing step.

We describe our approach as follows. Section 2 reviews how speaker content graphs are constructed, and Section 3 gives an overview of the clustering algorithms that we consider. We present the results of some initial experiments in Section 4 before proposing and validating further refinements in Section 5. Finally, Section 6 concludes this paper with a summary and discussion of future prospects.

This work was sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

2. SPEAKER CONTENT GRAPHS

The construction of a speaker content graph assumes that each node n in the graph corresponds to a single vector $\mathbf{m}$ from a speech signal. We define edge weights, or *affinities*, between two arbitrary nodes i and j via an affinity matrix $W = [W_{i,j}]$ where

$$W_{i,j} = \begin{cases} e^{-d^2(\mathbf{m}_i,\mathbf{m}_j)/\sigma^2} & \text{if an edge exists between } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The parameter σ controls the decay of the exponential function and, in our methods, $d(\cdot,\cdot)$ corresponds to the Euclidean distance between two speaker GMM supervectors as described in [4].

For the graph to remain sparse and still reflect the local neighborhood of a point $\mathbf{m}$, we connect $\mathbf{m}$ with only its top-K closest neighbors; i.e., connect $\mathbf{m}$ to some other node $\mathbf{m}_i$ only for the smallest K values of $d(\mathbf{m},\mathbf{m}_i)$ as $\mathbf{m}_i$ ranges the entire set of N vectors. Note that this construction implies the minimum degree of each node is K, but because the edge construction is done separately at each node, the degree of any particular node could be substantially larger than K. We will refer to K as a measure of a graph's edge *density*.

3. GRAPH CLUSTERING ALGORITHMS

Agglomerative Hierarchical Clustering (AHC) This simple, greedy algorithm is the predominant clustering approach used in speaker diarization systems [10]. Each node is initialized as its own cluster and iteratively merged with other clusters via some similarity metric – we found the unweighted group average affinity to perform best [11] – until some stopping criterion (e.g., BIC [7, 10], maximum distance [8], number of clusters, etc.) is met.

Modularity Optimization (CNM) The classical method and implementation proposed by Clauset, Newman, and Moore [12] seeks to optimize a graph's modularity, which is an estimate of the goodness of a partition based on a comparison between the graph at hand and a random null model [13]. Unfortunately, the resolution limit of such methods hindered this algorithm's performance [14]; we discuss these shortcomings in Section 4.

Spectral Clustering Spectral clustering can produce high-quality clusterings on small data sets, such as for speaker diarization of single conversations [15, 16]. However, these methods have limited scalability due to the required computational cost to obtain the eigenvectors of the data affinity matrix. Recent works have developed techniques to sparsify and simplify the procedure via the use of representative samples [17] or parallelization [18]. For this paper however, we simply apply the Ng-Jordan-Weiss (NJW) algorithm [19] in vanilla fashion and discuss its performance in Section 4.

Markov Clustering (MCL) [20] As summarized in [13], this algorithm converts a graph affinity matrix to a stochastic matrix by dividing the elements of each row by their sum and then iterates between two steps. During *expansion*, we compute an integer power of this matrix (usually a square), which yields the probability matrix of a random walk after that number of steps (e.g., 2). During *inflation*, each element of the matrix is raised to some power, α, artificially enhancing the probability of a random walker being trapped within a community. By solely iterating on the stochastic matrix, this method satisfies the Markov property, and we obtain clusters of separated communities upon convergence.

Infomap [21] The problem of finding the best cluster structure of a graph can be seen as the problem of optimally compressing its associated random walk sequence. The goal of Infomap is to arrive at a two-level description that exploits both the network's structure and the fact that a random walker is statistically likely to spend long periods of time within certain clusters of nodes. More specifically, we look for a module partition $\mathbf{M}$ (i.e., set of cluster assignments) of N nodes into m clusters that minimizes the following expected description length of a single step in a random walk on the graph:

$$L(\mathbf{M}) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p_{\circlearrowright}^{i} H(\mathcal{P}^{i}). \qquad (2)$$

This equation comprises two terms: first is the entropy of the movement between clusters, and second is the entropy of movements within clusters, both of which are weighted respectively by the frequency with which it occurs in the particular partitioning. The specifics are detailed in [21]; we provide this brief overview in preparation for our proposed refinements in Section 5.

Ultimately, Eqn. (2) serves as a criterion for a bottom-up agglomerative clustering search. The implementation provided by [21] uses Eqn. (2) to repeatedly merge the two clusters that give the largest decrease in description length until further merging gives an increase. Results are further refined using a simulated annealing approach, the specifics of which can again be found in [21].

4. PRELIMINARY EXPERIMENTS

We run the algorithms described in Section 3 following the default settings for MCL ($\alpha = 2$) and providing the number of clusters to both AHC and Spectral-NJW. There do exist a number of ways to detect the number of clusters from the affinity matrix [15, 16, 19, 22]; however, the computational cost of obtaining all the eigenvalues and eigenvectors of a 10,000-dimensional affinity matrix, albeit sparse, is non-trivial. We settle for computing just the largest eigenvalues and eigenvectors necessary [19].

4.1. Setup

To properly cross-validate performance, we evaluate on two sets of speaker content graphs - one built by combining the NIST SRE data from 2005 and 2006, and another built from 2008 and 2010 data. Both sets include both genders; more detailed information regarding the distribution of utterances per speaker is shown in Figure 1.

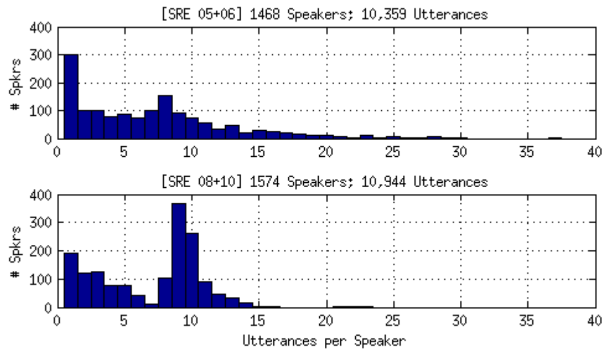

Fig. 1. *Distribution of the number of utterances per speaker on each test dataset (NIST SRE 05+06 and NIST SRE 08+10).*

Each speaker utterance is modeled using a MAP-adapted GMM supervector [23] and compared to other utterances via a Euclidean distance metric as described in [4]. Lastly, the global scaling parameter σ from Eqn. (1) is set to be $\sigma^2 = 0.5$; it was found empirically that the value of this parameter does not affect clustering performance. Each respective set consists of graphs of different edge densities, built from $K = \{2, 5, 10, 25, 50, 100\}$ nearest neighbors.

4.2. Evaluation Protocol

There exist a number of different metrics for evaluating cluster quality, including Precision and Recall, Normalized Mutual Information, F-score, B-cubed, et cetera [24]. We describe the one we chose below, which met our desire for a single number that summarizes the essentials and allows us to seamlessly compare performance across all algorithms and their parameters.

Let our r hypothesized clusters be indexed by i and our s true clusters be indexed by j. We evaluate our clustering output by considering all possible alignments that assign each hypothesized cluster i to exactly one true cluster j. Given such an alignment, say $i \leftrightarrow j$, we define cluster error as the number of elements in hypothesized cluster i whose true cluster is not actually j. Furthermore, any of the $|r - s|$ extra hypothesized or true clusters that did not receive an assignment are also counted as errors. Every possible alignment is considered, and the alignment that provides the smallest clustering error is used. In enforcing a one-to-one assignment of hypothesized-to-true clusters, we are able to summarize both the precision and recall of our clustering output. The procedure described above is equivalent to the evaluation procedure of "speaker confusion error" in the NIST Speaker Diarization task [25].

4.3. Initial Results

The results of our initial experiment are summarized in Figure 2. We can immediately see a systematic dependency between clustering performance and graph edge density. From Figure 1, it makes sense that our best results are obtained, across all algorithms, on the 5-NN and 10-NN graphs, since the 5-10 nearest neighbors of an utterance spoken by a particular speaker should ideally be the rest of the utterances spoken by that same speaker on average.

Although the performances of both AHC and Spectral-NJW are relatively consistent across graph densities, we should realize that

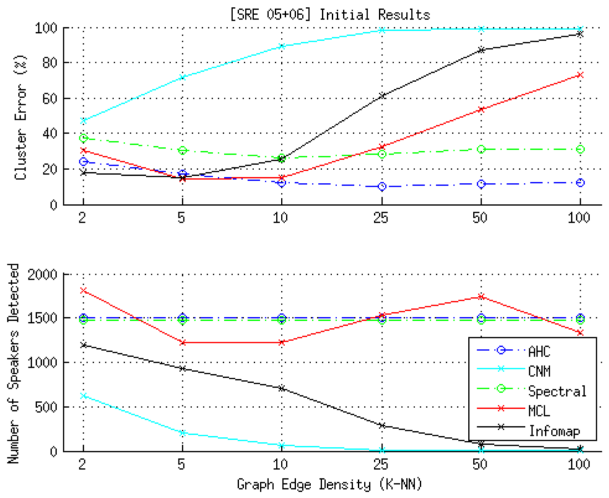

Fig. 2. *Initial results obtained on NIST SRE 05+06 data, using respective default/standard parameter settings for all algorithms considered. Top: Cluster Error as described in Section 4.2. Bottom: Number of speakers estimated by each algorithm; note AHC and Spectral-NJW were both given the number of speakers as input. The legend is common to both plots.*

the number of clusters was given as input to both algorithms. In the case of AHC, tuning the parameters of the various possible stopping criteria is beyond the scope of this paper. For Spectral-NJW, the number of clusters was provided to reduce the time spent computing eigenvalues and eigenvectors[1]. In the absence of a baseline method against which to evaluate our work, we use the results from AHC as an oracle benchmark for our refinements in Section 5.

The (Un)Importance of Edge Weight The densest graphs built using more nearest neighbors ($K = 50, 100$) exhibit worse clustering performance than their sparse counterparts. When the clustering algorithms see that an edge exists between two nodes (regardless of its weight), it assumes that their affinity is high relative to any other two nodes between which an edge does not exist (i.e., an affinity of 0). Thus, in the absence of information regarding the exact number of speakers (i.e., AHC and Spectral-NJW), a denser graph that has more edges by construction will be more likely to combine clusters that correspond to different speakers. We confirmed this hypothesis by running the same set of clustering experiments on unweighted versions of each graph and observing that the difference in clustering performance was negligible. Section 5 returns to this issue and proposes a way to improve performance on denser graphs.

The Limits of Resolution As mentioned in Section 3, we can attribute the poor performance of CNM to the resolution limit of modularity optimization methods [14]. Developing ways to overcome the resolution limit of CNM is beyond the scope of this paper; however, we acknowledge the development of other modularity optimization methods devoted to this issue [26]. From here, we proceed by limiting our consideration to the random walk-based algorithms of MCL and Infomap in our subsequent analyses.

[1]We should further note that the spectral clustering algorithm we use requires a pass of K-means clustering [19], whose complexity is NP-Hard.

5. SYSTEM REFINEMENTS

5.1. Parameter Tuning

MCL-α The input to MCL involves an inflation parameter, α, which is the power to which each element of the stochastic matrix is raised. An increase in α results in faster convergence and the detection of more speakers, while a decrease in α detects fewer speaker clusters. We also found that the optimal value of $\alpha = 2$ for graphs of all edge densities is not only independent of the density and the evaluation set, but is also the initial default value [20]. Figure 4 shows virtually no change in performance as a result of tuning α.

Infomap-λ Although the original formulation of Infomap in [21] involves no tuneable parameters, the minimization criterion presented in Eqn. (2) implicitly assigns equal weight to the between-cluster and within-cluster entropies. As such, we can introduce a parameter, λ, into the equation as follows:

$$L(\mathbf{M}) = q_{\curvearrowright} H(\mathcal{Q}) + \lambda \sum_{i=1}^{m} p_{\circlearrowright}^{i} H(\mathcal{P}^i). \tag{3}$$

The original Infomap corresponds to $\lambda = 1$. Letting $\lambda \to \infty$ increases our relative sensitivity to within-cluster entropy and yields more clusters that are smaller in size. Conversely, letting $\lambda \to 0$ favors larger and fewer clusters. Figure 3 shows the result of sweeping across a variety of different values of λ on both the 05+06 (solid lines and x's) and 08+10 (dash-dotted lines and o's) graphs.

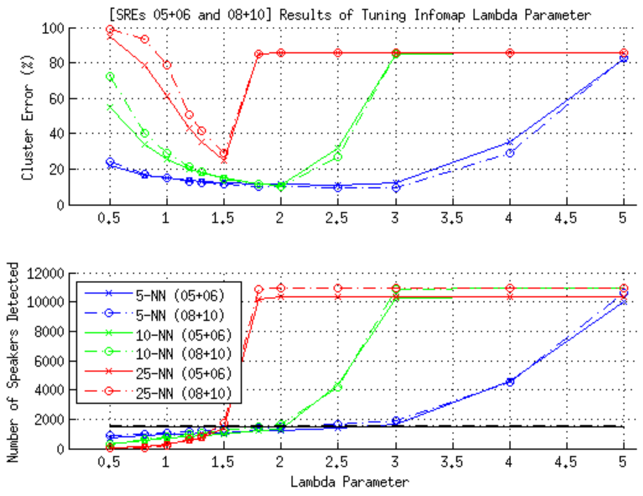

Fig. 3. *Results obtained on 05+06 and 08+10 evaluation sets for varying values of the Infomap-λ parameter. For clarity, we show only results from a subset of the graph densities, but the trends are similar for other graphs. The legend is common to both plots.*

For different graph edge densities there exist different values of λ that cause every cluster to be a singleton; furthermore, the optimal value of λ depends strongly on the edge density. But because the trends are consistent across evaluation sets, it is plausible to pick these values of λ *a priori*, as though one of the evaluation sets were a development set and the other a test set[2]. Upon doing so, Fig-

[2]Without further validation, we cannot state this as a claim in general. Indeed, the respective optimality of these algorithms' parameters may be an incident of our datasets (SREs 05+06 and 08+10) being too similar in degree distribution (i.e., Figure 1).

ure 4 shows the result of Infomap-original and Infomap-λ alongside other methods (MCL-original and MCL-α) considered in this paper. For each edge density $K \in \{2, 5, 10, 25, 50, 100\}$, λ_K (as well as α_K) is chosen as the value that yields the best clustering error on the 05+06 graph of edge density K as seen in Figure 3. We can see that introducing and optimizing Infomap-λ results in clustering performance (on 08+10 data) competitive with both MCL and MCL-α.

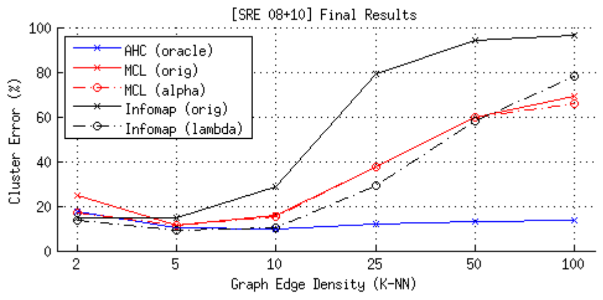

Fig. 4. *Final cluster errors obtained on SRE 08+10 data, using both default parameters (solid lines and x's) and graph density-optimized parameters for Infomap-λ and MCL-α (dash-dotted lines and o's).*

5.2. Local Node Refinements

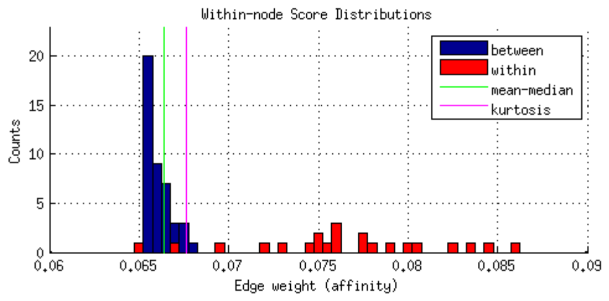

Fig. 5. *Histogram of within- and between-speaker score distributions as well as the cutoff thresholds discussed in Section 5.2.*

In Section 4.3, we demonstrated that information regarding edge weight magnitude has little impact on clustering performance. To facilitate a more in-depth analysis, Figure 5 presents the top 100 edge weights of an arbitrary node/utterance A produced by speaker s_A. These edge weights are separated into two different histograms: red "within-speaker" scores and blue "between-speaker" scores. The combined score distribution, which is what the clustering algorithms see, has a right skew. Assuming, per the speaker recognition literature, that both within- and between-speaker scores can be modeled using respective Gaussian distributions [27], we can use simple measures of symmetry and kurtosis to arrive at the following heuristic to prune away between-speaker edges.

Let Z_A denote the combined distribution of scores for some node A. We keep the subset of scores Z_A^+, or edges, that are greater than some threshold θ_{mm} (i.e., $Z_A^+ = \{z \in Z_A | z > \theta_{\text{mm}}\}$), where θ_{mm} is the largest value such that for the subset of scores $Z_A^- = \{z \in Z_A | z \leq \theta_{\text{mm}}\}$, $\text{mean}(Z_A^-) \leq \text{median}(Z_A^-)$. This method assumes that the mean should be greater than the median in a combined score distribution with a right skew, but without the tail of within-speaker scores, the remaining between-speaker score distribution should be reasonably symmetric.

Taking the assumption of between-speaker score Gaussianity a step further, we introduce kurtosis into our local-node pruning. In this case, we choose θ_{kurt} to be the largest score value such that $\text{kurtosis}(Z_A^-) \leq 3$, where 3 is the kurtosis of a normal distribution. Figure 5 shows the cutoff found by kurtosis in magenta, as well as the cutoff, in green, found by the mean-median method above.

It was found that the combination of methods worked best; in particular, the results shown in Figure 6 were obtained using the threshold $\tilde{\theta} = \max\{\theta_{\text{mm}}, \theta_{\text{kurt}}\}$. An edge was pruned away if either node in the edge-pair deemed the connection unnecessary. The resulting average degree, or number of edges, for each (post-pruned) node is shown on the x-axis. We can see that the pruning improves performance for the denser graphs (i.e., 25-, 50-, and 100-NN), but can hurt slightly for sparser graphs. This makes sense, as our pruning methods assume a heavy presence of between-speaker scores, and a sparser graph would contain fewer of these samples. These results were generated using both MCL and Infomap with default parameter settings. With this refinement method, we are able to attain clustering performance equal to that of our AHC oracle benchmark.

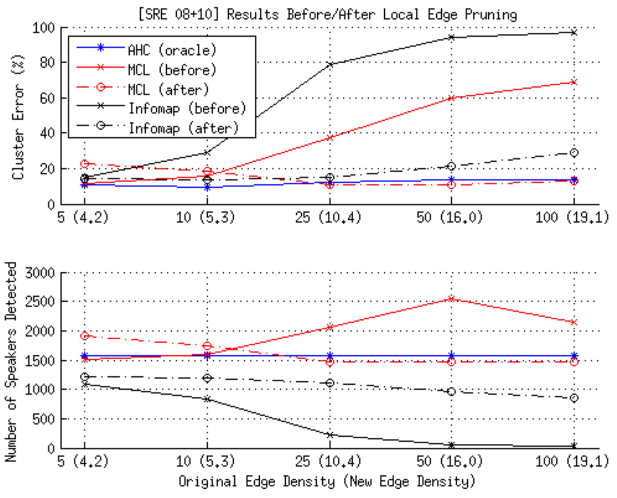

Fig. 6. *08+10 results obtained after pruning away scores of a 100-NN graph using the heuristics described in Section 5.2. Both original and new graph densities are labeled on the x-axis.*

6. CONCLUSION

In this paper, we have surveyed a number of methods to perform tractable, large-scale clustering on sparse speaker content graphs containing over 10,000 nodes. In obtaining clustering error rates as low as 10%, AHC provides the best results across graphs of all edge densities; however, such performance is contingent on the number of clusters provided as input, and it was outside the current research scope to address less-supervised stopping criteria.

The random walk algorithms, MCL and Infomap, show promising results. MCL worked consistently and performed well without the need for any parameter tuning. And after introducing a λ parameter into the optimization, Infomap could be tuned to work at least as well as MCL. Lastly, we proposed techniques to better utilize the information provided in denser graphs, resulting in clustering performance competitive with that of our AHC oracle benchmark.

Acknowledgments We would like to thank Najim Dehak and Jim Glass for their helpful insights in the development of this work.

7. REFERENCES

[1] William M. Campbell and Zahi Karam, "Simple and efficient speaker comparison using approximate KL divergence," in *Proceedings of Interspeech*, 2010.

[2] Patrick Kenny, Pierre Ouellet, Najim Dehak, Vishwa Gupta, and Pierre Dumouchel, "A study of inter-speaker variability in speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, 2008.

[3] Najim Dehak, Patrick Kenny, Reda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech and Language Processing*, 2011.

[4] Zahi Karam and William M. Campbell, "Graph embedding for speaker recognition," in *Proc. Interspeech*, 2010, pp. 2742–2745.

[5] Zahi Karam, William M. Campbell, and Najim Dehak, "Graph relational features for speaker recognition and mining," in *IEEE Statistical Signal Processing Workshop*, 2011, pp. 525–528.

[6] William M. Campbell and Elliot Singer, "Query-by-example using speaker content graphs," in *Proc. Interspeech*, 2012.

[7] David A. van Leeuwen, "Speaker linking in large data sets," in *Proc. Odyssey*, 2010.

[8] Marijn Huijbregts and David van Leeuwen, "Large scale speaker diarization for long recordings and small collections," *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[9] Ullas Gargi, Wenjun Lu, Vahab Mirrokni, and Sangho Yoon, "Large-scale community detection on youtube for topic discovery and exploration," in *Proc. of the Fifth international AAAI Conference on Weblogs and Social Media*, 2011, pp. 486–489.

[10] Sue E. Tranter and Douglas A. Reynolds, "An overview of automatic speaker diarization systems," *IEEE Transactions on Audio, Speech, and Language Processing*, 2006.

[11] R. Sokal and C. Michener, "A statistical method for evaluating systematic relationships," *University of Kansas Science Bulletin*, 1958.

[12] Aaron Clauset, Mark E. J. Newman, and Cristopher Moore, "finding community structure in very large networks," *Physical Review E*, 2004.

[13] Andrea Lancichinetti and Santo Fortunato, "Community detection algorithms: a comparative analysis," *Physical Review E*, 2009.

[14] Santo Fortunato and Marc Barthelemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, 2007.

[15] Huazhong Ning, Ming Liu, Hao Tang, and Thomas S. Huang, "A spectral clustering approach to speaker diarization," in *Proc. ICSLP*, 2006.

[16] Stephen Shum, Najim Dehak, and Jim Glass, "On the use of spectral and iterative methods for speaker diarization," in *Proc. Interspeech*, 2012.

[17] Donghui Yan, Ling Huang, and Michael I. Jordan, "Fast approximate spectral clustering," in *15th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2009.

[18] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y. Chang, "Parallel spectral clustering in distributed systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[19] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, 2001, pp. 849–856.

[20] Stijn van Dongen, *Graph Clustering by Flow Simulation*, Ph.D. thesis, University of Utrecht, May 2000.

[21] Martin Rosvall and Carl T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, 2008.

[22] Lihi Zelnik-Manor and Pietro Perona, "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems*, 2004.

[23] William M. Campbell, Douglas E. Sturim, Douglas A. Reynolds, and Alex Solomonoff, "SVM based speaker verification using a GMM supervector kernel and NAP variability compensation," in *Proceedings of ICASSP*, 2006, pp. I–97–I–100.

[24] Enrique Amigo, Julio Gonzalo, Javier Artiles, and Felisa Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information Retrieval*, 2009.

[25] NIST, "Diarization error rate (DER) scoring code," 2006, www.nist.gov/speech/tests/rt/2006-spring/code/md-eval-v21.pl.

[26] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics*, 2008.

[27] Douglas Reynolds, Thomas Quatieri, and Robert Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, 2000.