An Evaluation of Graph Clustering Methods for Unsupervised Term Discovery

Vince Lyzinski, Gregory Sell, Aren Jansen

Human Language Technology Center of Excellence & Center for Language and Speech Processing
The Johns Hopkins University, Baltimore, MD USA
{vlyzins1, gsell, aren}@jhu.edu

Abstract

Unsupervised term discovery (UTD) is the task of automatically identifying the repeated words and phrases in a collection of speech audio without relying on any language-specific resources. While the solution space for the task is far from fully explored, the dominant approach to date decomposes the discovery problem into two steps, where (i) segmental dynamic time warping is used to search the speech audio for repeated acoustic patterns, and (ii) these individual repetitions are partitioned into word/phrase categories using graph clustering. In this paper, we perform an unprecedented evaluation of a wide range of advanced graph clustering methods for the UTD task. We conduct our study in the evaluation framework of the Zero Resource Speech Challenge. We find that, for a range of features and languages, modularity-based clustering improves UTD performance most consistently, often by a wide margin. When paired with out-of-language deep neural net bottleneck features, we find performance near that of a high-resource UTD system.

Index Terms: Unsupervised term discovery, graph clustering, zero resource speech challenge

1. Introduction

The lack of transcribed speech for a given language or domain (the so-called zero resource setting) precludes both the training of highly accurate recognizers as well as any downstream technologies that depend on lexical or phonetic tokenizations. The task of unsupervised term discovery (UTD) attempts to sidestep the need for word transcripts and pronunciation dictionaries by automatically discovering clusters of repeated words and phrases in untranscribed audio [1, 2, 3, 4, 5, 6]. While the resulting tokenizations do not explicitly carry meaning (each type receives a unique generic identifier), UTD has been demonstrated to be a useful preprocessing step for downstream summarization [7, 8], topic identification [9, 10, 8], spoken document retrieval [11, 12], and speaker recognition [13] technologies. Moreover, recent studies have demonstrated the value of automatically discovered words as weak supervision for acoustic model training [14, 15, 16, 17].

Park and Glass [1] introduced the UTD problem and proposed a first solution, which involved an $O(n^2)$ search for closely repeating acoustic patterns using their segmental dynamic time warping (S-DTW) algorithm. The discovered repetitions were subsequently used to construct a graph, where nodes represented intervals of speech and edges reflect which intervals were acoustically similar. Finally, the resulting word and phrase categories (which we refer to as pseudoterms, due to their indefinite nature) were identified using a fast graph clustering method proposed by Newman [18].

Since this original work, several improvements were proposed, including methods to improve scalability of the S-DTW search [19, 4] and word discriminability of the segment acoustic similarity metric [20]. However, none of these subsequent efforts have focused on improving the graph clustering procedure, with only Newman's algorithm and simple connected components clustering having been considered to date. Moreover, while the improvements in computational efficiency of S-DTW allowed processing of much larger corpora (100s of hours), this leads to even bigger graphs that make clustering algorithm scalability an even more important factor.

With these considerations in mind, this paper presents a comprehensive evaluation of several state-of-the-art graph clustering algorithms for UTD systems. Of critical importance is algorithm scalability, as even small corpora can generate graphs with millions of nodes and tens of millions of edges. In addition, the unsupervised nature of the problem also favors algorithms with fewer parameters to tune since in real applications any validation must be performed manually. We use the Zero Resource Speech Challenge (Track 2) framework [21], which includes evaluation sets for English and Xitsonga, the latter of which serves as a veritable zero resource test language. We consider the both the evaluation metrics used in [4], as well as the comprehensive array of UTD performance metrics provided by the challenge. This enables not only an internal comparison of the clustering methods with a fixed graph construction methodology, but also allows external comparisons to other UTD systems evaluated by other challenge participants. We begin with a detailed description of our graph construction methodology.

2. Graph Construction

All experiments in our study employ the scalable S-DTW-based term discovery system procedure presented in [4]. This system takes as input a collection of speech documents and produces a list of P scored interval pairs $\mathcal{P} = \{(x_i^1, x_i^2, s_i)\}_{i=1}^P$, where $s_i \in [0,1]$ is a normalized acoustic similarity between the intervals and each interval x_i^j is specified by a triple (f_i^j, a_i^j, b_i^j) indicating the audio file f_i^j, start time a_i^j, and end time b_i^j. Once this output has been obtained, the next step is to transform $\mathcal{P}$ into a weighted undirected graph $G = (V, E)$, where V is the vertex set and E is the weighted edge set.

The goal of our graph clustering procedure is to partition the individual intervals in $\mathcal{P}$ into clusters that contain multiple utterances of (ideally) a single word or phrase type and where each type is present in as few clusters as possible. Thus, for each of the individual intervals x_i^j that comprises half of a pair in $\mathcal{P}$ we define a corresponding vertex in v_i^j in V, such that $|V| = 2 \cdot P$. Now, if a given term occurs k times in the speech, the S-DTW procedure will produce as many as $k(k-1)/2$ matches that generate as many as $k(k-1)$ vertices. To successfully

cluster all of these vertices into a single pseudoterm cluster, we must introduce two classes of edges: *acoustic* and *overlap*.

Acoustic edges result from DTW similarity of distinct regions of speech and are derived from $\mathcal{P}$. Specifically, for each interval pair in $(x_i^1, x_i^2, s_i) \in \mathcal{P}$, we introduce an edge between vertices v_i^1 and v_i^2 of weight $1/(1+\exp[-(s_i-\tau_a)/20])$, where τ_a is a free parameter. At this point, our graph has a ladder structure, consisting of P connected components of 2 vertices each. We then further augment the graph by introducing a set of overlap edges that indicate to what degree two intervals overlap in the same source audio file. Formally, we insert one edge between vertices v_i^j and v_k^l if the corresponding intervals (f_i^j, a_i^j, b_i^j) and (f_k^l, a_k^l, b_k^l) are contained in the same file (i.e., $f_i^j = f_k^l$) and have non-zero overlap in time. The weight of each overlap edge is defined as $1/(1+\exp[-(F(x_i^j, x_k^l)-\tau_o)/20])$, where τ_o is a free parameter and the fraction overlap $F(x_i^j, x_k^l)$ can be efficiently computed as $\max(0, 1 - r(x_i^j, x_k^l))$, where

$$r(x_i^j, x_k^l) = \frac{b_i^j - a_i^j + b_k^l - a_k^l}{\max(b_i^j, b_k^l) - \min(a_i^j, a_k^l)}.$$

Note that since the two types of edges are measuring different types of vertex similarity with distinct distributional properties, it is necessary to treat them separately in the downstream graph clustering procedures. This is handled with separate τ_a and τ_o parameters. For each acoustic front-end we consider in our experiments, we create one master graph for each evaluation language by applying a very conservative threshold (i.e. far below considered values for τ_a and τ_o) on both the acoustic and overlap edges solely for the purpose of keeping the graph sizes manageable. Additionally, we only keep intervals of at least 0.5 seconds in duration.

3. Graph Clustering Methods

In addition to simply thresholding the graph edges (at 0.5) and using the connected components (referred to as ConnComp below), we consider four state-of-the-art (both in efficacy and scalability) graph clustering procedures: the modularity-based algorithms of [22] and [23], the label propagation algorithm of [24], and the information flow based algorithm of [25].

3.1. Modularity-Based Clustering

Modularity is a widely applied metric for measuring the performance of graph clustering algorithms [26, 27, 28]. Heuristically, the modularity of a clustering measures the fraction of edge weights that fall within clusters as opposed to across clusters. More precisely, given a weighted graph $G = (V, E)$ and a clustering of V into m components $\{c_1, c_2, \ldots, c_m\}$, the modularity (suitably normalized below to take a value in $[-1, 1]$) is defined as

$$Q = \frac{1}{2w} \sum_{i,j} \left[E_{i,j} - \frac{w_i w_j}{2w} \right] \delta(c_i, c_j), \tag{1}$$

where $w_i = \sum_j E_{i,j}$ is the sum of the weights of edges incident to vertex i, $w = \frac{1}{2} \sum_{i,j} E_{i,j}$ is the total weight of the network, c_i is the cluster component vertex v_i is assigned to, and $\delta(c_i, c_j) = 1$ if $c_i = c_j$ and 0 otherwise. If the modularity is close to 1 (resp., -1), then a significantly larger fraction of the total edge weights are within (resp., between) clusters as opposed to between (resp., within) clusters. If the fraction of edge weights within clusters is what we would expect from a random network with the given vertex degrees, the modularity will be close to 0. While exactly optimizing modularity is NP-hard [28], numerous highly scalable algorithms have been proposed in the literature to find a clustering that approximately optimizes Eq. 1. We consider two of the most prominent modularity-based clustering methods: the Louvain algorithm [22] and the algorithm of Clauset, Newman, and Moore [23, 29] (referred to as FastGreedy in our experiments).

The Louvain algorithm proceeds in two iterated phases. Phase one begins with every vertex assigned to a separate cluster. The algorithm then sequentially considers each vertex $v_i \in V$ and, for each $v_j \in N_G(v_i) := \{v_k \in V | E_{i,k} \neq 0\}$, evaluates the gain in modularity achieved by removing v_i from its present cluster and putting it into the cluster containing v_j. Vertex v_i is then placed in the cluster which maximizes this gain, or kept in its present cluster if no gain is possible. This process is then iteratively and sequentially applied on all vertices until no further increases in modularity can be achieved. In the second phase, the resulting clusters are collapsed to single vertices, with edge weights between vertices computed by summing the edge weights between the two pre-collapsed clusters. The two phases are then iterated on the collapsed graphs until the vertices all belong to a single cluster. The resulting clustering is essentially hierarchical, and we use the clustering in the hierarchy that maximizes modularity in G to cluster our graph. The Louvain algorithm is unsupervised and very fast, feasibly running on extremely large graphs (order $\approx 10^9$ vertices in [22]).

The clustering algorithm of Clauset, Newman, and Moore [23, 29] is a more efficient implementation of Newman's algorithm [18] used by Park and Glass [1], but proceeds similarly. Beginning with every vertex assigned to a separate cluster, the algorithm efficiently calculates the change in modularity that would result from merging each pair of clusters. The pair whose merging would result in the maximum modularity are then merged, and the algorithm is iterated on the resulting clusters. Again, the output of the algorithm is essentially a hierarchical clustering, and we use the clustering in the hierarchy that achieves the maximum modularity in G. The algorithm is fully unsupervised and very fast; indeed, in the setting of many sparse, real-world (unweighted) networks, the algorithm has a nearly linear run time $O(|E| \log^2 |V|)$ [23] (compared to the $O(|V| \cdot |E|)$ runtime of [18]).

3.2. Label Propagation and Information Flow

We also explore two state-of-the-art clustering algorithms based on non-modularity clustering heuristics. The first is a weighted variant of the label propagation algorithm of [24], which we refer to as LabelProp below. The heuristic behind the algorithm is very simple. If vertex v_i has neighbors $N_G(v_i) = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$, with corresponding cluster labels $c_1, c_2, \ldots, c_k$, then v_i determines its cluster assignment by maximizing

$$\sum_{j=1}^{k} \delta(c_j, c) E_{i,i_j} \tag{2}$$

over all possible cluster labels c. With this in mind, the algorithm proceeds as follows. Initialize with every vertex being assigned to a separate unique cluster. Iteratively and sequentially over all vertices, each node updates its cluster assignment by maximizing Eq. 2 over cluster labels c. The cluster assign-

ment updates are performed asynchronously, and the order in which the vertices' cluster memberships are updated in each iteration of the algorithm is random. In the unweighted setting, each iteration of the algorithm has runtime $O(|E|)$, and excellent performance is often achieved with few iterations [24]. Label propagation is fully unsupervised.

Lastly, we consider InfoMap, the information flow based algorithm of [25, 30]. Heuristically, the InfoMap algorithm exploits the duality between data compression and pattern/structure extraction to cluster the vertices in the graph by minimizing the description length of information flows on the graph. More precisely, given a clustering $\mathcal{C}$ of the vertices of the graph into m clusters, the algorithm uses a two-level coding scheme to efficiently encode the trajectory of a random walk on the graph as follows. Each cluster is given a unique code name and different Huffman coding schemes [31] are used to name the vertices within each cluster. Each cluster is then assigned a unique exit code to indicate when the random walk exits the cluster. The average description length of a step of the random walk on the network can then be written as

$$L(P) = qH(\mathcal{C}) + \sum_{i=1}^{m} p_i H(\mathcal{C}_i), \tag{3}$$

where q is the probability that the random walk moves to a different cluster at any given step, $H(\mathcal{C})$ is the entropy of the cluster code names, $H(\mathcal{C}_i)$ is the entropy of the within cluster code names (including the exit code), and p_i is the proportion of within cluster moves that occur within cluster i, plus the probability of exiting cluster i. The InfoMap algorithm employs a greedy search algorithm and simulated annealing to approximately minimize Eq. 3 over all clusterings $\mathcal{C}$. This flow based clustering procedure is also fully unsupervised as implemented and scales well to very large graphs.

4. Experiments

The above clustering methods were evaluated within the Zero Resource Speech Challenge (Track 2) [21], which includes two separate speech corpora. First, the challenge uses a subset of the Buckeye corpus, consisting of 5 hours of English conversational speech. This data is microphone recorded and sampled at 16 kHz. The second corpus consists of 2.5 hours of microphone recordings of Xitsonga (a South African language) prompted speech, also sampled at 16 kHz.

4.1. Features

For each language, three sets of features were considered: perceptual linear prediction (PLP), short-time frequency domain linear prediction (FDLPS), and bottleneck features (BNF). The 39-dimensional PLP features (13 + deltas + double deltas) were also used for the official challenge baseline systems, while the 15-dimensional FDLPS features (5 + deltas + double deltas) filter spectral detail encoded in higher-order cepstral coefficients to improve speaker independence as demonstrated in [32]. Both PLP and FDLPS features are raw acoustic features and do not involve any language resources or training.

The BNFs were computed from a 5-layer deep neural network (DNN) trained on 1,500 hours of out-of-domain Fisher English telephone speech. Each layer included 5,000 units with 2-norm maxout nonlinearities, and the final softmax output layer targeted 7,600 clustered context-dependent HMM states (senones). The initial input to the net was a 9-frame context window of MFCCs, and the resulting 60-dimensional BNFs should encode the acoustic structure relevant to English. While the BNFs are therefore utilizing in-language (though channel-mismatched) data for English (Buckeye), the Xitsonga results use only out-of-language resources for BNFs, which remains a legitimate zero-resource scenario.

Table 1: Average runtime (in seconds) on the FDLPS Buckeye graph (136,324 vertices, 578,329 edges). Averages are computed over all edge weightings considered.

Algorithm	`igraph` Routine	Runtime (s)
FastGreedy	`cluster_fast_greedy`	27.9
InfoMap	`cluster_infomap`	293.9
LabelProp	`cluster_label_prop`	3.3
Louvain	`cluster_louvain`	4.1

4.2. Metrics

We consider the performance of these clustering methods in light of several UTD metrics. First, we evaluate according to performance on the subset of long-duration words, which are assumed to be primarily content words that are of higher importance for downstream applications such as topic modeling and keyword discovery [4]. The systems considered in this work were designed for these tasks, and as a result, evaluating on content words is the most appropriate metric. Using as targets all same-type pairs of reference words in the evaluation set that are at least 0.5 seconds, we measured precision and recall of the same-type pairs predicted by our clustering algorithms. The performance is summarized using the F-score.

Next, we evaluated according to the Zero Resource Speech Challenge metrics [21], which considers discovery of all phone n-grams of varying length (down to order 3). Performance is measured in terms of normalized edit distance (NED); total coverage; precision, recall, and F-score for the n-gram matching task; and, type, token, and boundary metrics for word segmentation tasks. Note that since our system only hypothesizes repetitions of at least 0.5 seconds, a recall ceiling is implicit. Moreover, since our UTD system does not provide a single comprehensive segmentation, the "matching" metrics are the primary focus. We report all challenge metrics for completeness.

4.3. Results

4.3.1. Runtimes

For all clustering algorithms evaluated, we used the efficient implementations provided in the open-source network analysis package `igraph` [33]. Table 1 lists the average processing time for each of the methods applied to the largest graph we generated in our experimentation. The flow-based InfoMap algorithm is by far the slowest, with runtimes an order of magnitude larger than the nearest competitor. The modularity-based Louvain algorithm, which we will see is of the most accurate methods, nearly matches the fastest runtime measured.

4.3.2. Results on Content Words

Results for the content-word F-scores are shown in Table 2. Optimal τ_a and τ_b parameters for Buckeye were selected based on performance on the Xitsonga corpus, and vice versa for the Xitsonga system, so the parameter selections were fair. It is immediately evident from these scores that the modularity-based clustering methods (Louvain and FastGreedy) offer improvements over ConnComp in all cases, and provide the top perfor-

Table 2: Content-word F-scores (%) for the Buckeye and Xitsonga corpora with different feature types. Buckeye-BNF scores are italicized to represent the use of in-language training data in this case.

	Buckeye			Xitsonga		
	PLP	FDLPS	*BNF*	PLP	FDLPS	BNF
ConnComp	16.3	19.9	*42.7*	2.9	13.5	25.7
InfoMap	15.0	24.3	*26.1*	2.8	**21.4**	11.6
LabelProp	13.0	21.1	*19.7*	2.2	16.1	7.9
Louvain	**23.3**	26.6	*46.7*	**4.9**	18.0	**40.0**
FastGreedy	23.3	**28.1**	***47.0***	4.3	17.2	29.6

Table 3: Zero Resource Speech Challenge evaluation systems (CC=ConnComp, FG=FastGreedy, L=Louvain, plus parameters τ_a and τ_o) and scores (%) for Buckeye (top) and Xitsonga (bottom). BNF scores are italicized to indicated the use of in-language supervision.

	Clustering Method	NLP		Matching			Type			Token			Boundary		
		NED	Cov.	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	-	21.9	16.3	39.4	1.6	3.1	6.2	1.9	2.9	5.5	0.4	0.8	44.1	4.7	8.6
Topline	-	0	100	98.3	18.5	31.1	50.3	56.2	53.1	68.2	60.8	64.3	88.4	86.7	87.5
PLP	CC, 0.87, 0.8	0.773	25.5	14.6	2.3	4.0	4.7	2.5	3.3	4.2	0.6	1.0	39.6	7.5	12.7
FDLPS	CC, 0.9, 1.0	0.612	80.2	6.5	3.5	4.6	3.1	9.2	4.6	2.4	3.5	2.8	35.4	38.5	36.9
BNF	FG, 0.9, 1.0	*0.364*	*46.7*	*12.9*	*5.1*	*7.2*	*2.3*	*2.9*	*2.6*	*1.9*	*0.7*	*1.0*	*31.7*	*14.2*	*19.6*

	Clustering Method	NLP		Matching			Type			Token			Boundary		
		NED	Cov.	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	-	12	16.2	69.1	0.3	0.5	3.2	1.4	2	2.6	0.5	0.8	22.3	5.6	8.9
Topline	-	0	100	100	6.8	12.7	15.1	18.1	16.5	34.1	39.7	40.4	66.6	91.9	77.2
PLP	FG, 0.87, 0.9	36.1	30.2	30.6	0.6	1.2	3.0	2.7	2.8	2.0	0.9	1.2	19.4	11.2	14.2
FDLPS	CC, 0.9, 1.0	43.2	89.4	21.2	3.8	6.5	4.9	18.8	7.8	2.2	12.6	3.8	18.8	64.0	29.0
BNF	L, 0.93, 1.0	34.1	67.6	13.3	7.4	9.5	2.6	6.0	3.6	1.5	2.3	2.0	14.8	29.5	19.7

mance in all cases except Xitsonga with FDLPS features. Between the two languages, we notice that the PLP performance is a great deal worse on Xitsonga than English, while performance is on a more similar order between the languages for FDLPS and BNF.

For Buckeye, it is not at all surprising that BNFs give the best overall performance, considering the use of copious in-language training data. Moreover, the majority of the improvements come from the features themselves, as there is very little difference between ConnComp, Louvain, and FastGreedy scores. However, for Xitsonga, the use of Louvain offers a large improvement over the already relatively high ConnComp score. The 40% F-score usings BNFs with Louvain falls only 14% relative behind the best performance of in-language BNFs on Buckeye. The BNFs utilize no in-language training data for Xitsonga, and so this result combining the BNFs with modularity clustering is extremely encouraging.

Finally, we note that even though the parameters for these systems were selected fairly based on out-of-language performance, oracle selections were separately analyzed. In the vast majority of cases, the fairly-selected parameters performed at or near the level of oracle-selected parameters. The overall stability between languages is encouraging for realizing these gains in zero-resource application settings.

4.3.3. Official Zero Resource Speech Challenge Results

The official Zero Resource Speech Challenge evaluation metrics are shown in Table 3 for Buckeye (top) and Xitsonga (bottom). These results are also shown in comparison to the performance of the official baseline (a UTD system with PLP featues and connect component clustering) and topline (a Bayesian word segmenter [32] applied to reference phonetic transcripts) for each corpus. As was the case with the content-word scores, τ_a and τ_o parameters were selected according to performance on the opposite corpus, in this case based on the matching F-score. In both cases, the general patterns are the same as seen in Table 2, in that FDLPS performance is better than PLP, and BNF is best overall (in terms of matching F-score). In both cases, these improvements appear to be largely related to an increase in recall. Also similar to the content-word score analysis, these performances were found to be stable compared to oracle selection. Finally, the BNF results are again very encouraging for the Xitsonga corpus. In this case, the matching F-score of the BNF system (which uses Louvain) has recovered over 75% of the performance gap between the topline and baseline.

5. Conclusions

Several UTD systems were evaluated both in terms of content words and the full set of phone n-grams, and several conclusions can be drawn from these experiments. First, FDLPS features are a preferable to PLP as an acoustic feature for UTD. However, the data-driven modeling in BNFs trained on out-of-language speech are superior to either acoustic feature for this task, and therefore appear to be the optimal choice. It was also consistently the case that clustering the identified candidate terms can be significantly improved by utilizing Louvain or FastGreedy, both modularity-based methods. Given its efficiency, Louvain is the optimal combination of speed and accuracy. In the case of BNFs on Xitsonga data, the resulting system was near the performance of an in-language system on Buckeye. Experiments also showed that optimal parameter and feature selection for these methods was relatively stable across languages, which is essential for use in true zero resource scenarios.

6. Acknowledgments

The authors thank Daniel Garcia-Romero of Johns Hopkins University for providing the neural network for BNF extraction.

7. References

[1] A. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *IEEE T-ASLP*, vol. 16, no. 1, pp. 186–197, 2008.

[2] V. Stouten, K. Demuynck *et al.*, "Discovering phone patterns in spoken utterances by non-negative matrix factorization," *Signal Processing Letters, IEEE*, vol. 15, pp. 131–134, 2008.

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

[4] A. Jansen and B. V. Durme, "Efficient spoken term discovery using randomized algorithms," in *Proc. ASRU*, 2011.

[5] O. Räsänen, "A computational model of word segmentation from continuous speech using transitional probabilities of atomic acoustic events," *Cognition*, vol. 120, no. 2, pp. 149–176, 2011.

[6] N. Vanhainen and G. Salvi, "Pattern discovery in continuous speech using block diagonal infinite HMM," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3719–3723.

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

[8] M. Siu, H. Gish, A. Chan, W. Belfield, and S. Lowe, "Unsupervised training of an hmm-based self-organizing unit recognizer with applications to topic classification and keyword discovery," *Computer Speech & Language*, vol. 28, no. 1, pp. 210–223, 2014.

[9] M. Dredze, A. Jansen, G. Coppersmith, and K. Church, "NLP on spoken documents without ASR," in *Proc. of EMNLP*, 2010.

[10] T. J. Hazen, M.-H. Siu, H. Gish, S. Lowe, and A. Chan, "Topic modeling for spoken documents using only phonetic information," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 395–400.

[11] J. White, D. Oard, A. Jansen, J. Paik, and R. Sankepally, "Using zero-resource spoken term discovery for ranked retrieval," in *Proc. of NAACL-HLT*, 2015.

[12] J. Wintrode, G. Sell, A. Jansen, M. Fox, D. Garcia-Romero, and A. McCree, "Content-based recommender systems for spoken documents," in *Proc. ICASSP*, 2015.

[13] A. Jansen, D. Garcia-Romero, P. Clark, and J. Hernandez-Cordero, "Unsupervised idiolect discovery for speaker recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1675–1679.

[14] A. Jansen and K. Church, "Towards unsupervised training of speaker independent acoustic models," in *Interspeech*, 2011.

[15] A. Jansen, S. Thomas, and H. Hermansky, "Weak top-down constraints for unsupervised acoustic model training," in *IEEE ICASSP*, 2013.

[16] G. S. T. Schatz and E. Dupoux, "Phonetics embedding learning with side information," in *Proc. SLT*, 2014.

[17] H. Kamper, M. Elsner, A. Jansen, and S. Goldwater, "Unsupervised neural network based feature extraction using weak top-down constraints." in *Proc. ICASSP*. IEEE, 2015.

[18] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.

[19] Y. Zhang and J. R. Glass, "An inner-product lower-bound estimate for dynamic time warping," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5660–5663.

[20] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 410–415.

[21] M. Versteegh *et al.*, "The zero resource speech challenge 2015," in *Proc. of Interspeech*, 2015.

[22] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[23] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[24] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.

[25] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[26] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[27] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[28] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 2, pp. 172–188, 2008.

[29] M. E. J. Newman, "Analysis of weighted networks," *Physical Review E*, vol. 70, no. 5, p. 056131, 2004.

[30] M. Rosvall, D. Axelsson, and C. T. Bergstrom, "The map equation," *The European Physical Journal-Special Topics*, vol. 178, no. 1, pp. 13–23, 2009.

[31] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[32] A. Jansen, E. Dupoux, S. Goldwater, M. Johnson, S. Khudanpur, K. Church, N. Feldman, H. Hermansky, F. Metze, R. C. Rose *et al.*, "A summary of the 2012 JHU CLSP Workshop on zero resource speech technologies and models of early language acquisition." in *ICASSP*, 2013, pp. 8111–8115.

[33] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal, Complex Systems*, vol. 1695, no. 5, pp. 1–9, 2006.