Real-Time Keyword Extraction from Conversations[*]

Polykarpos Meladianos
École Polytechnique & AUEB
pmeladianos@aueb.gr

Antoine J.-P. Tixier
École Polytechnique
antoine.tixier-1@colorado.edu

Giannis Nikolentzos
École Polytechnique & AUEB
nikolentzos@aueb.gr

Michalis Vazirgiannis
École Polytechnique
mvazirg@lix.polytechnique.fr

Abstract

We introduce a novel, fully unsupervised method to extract keywords from meeting speech in *real-time*. Our approach represents text as a word co-occurrence network and leverages the k-core graph decomposition algorithm and properties of submodular functions. We outperform multiple baselines in a real-time scenario emulated from the AMI and ICSI meeting corpora. Evaluation is conducted against both extractive and abstractive gold standard using two standard performance metrics and a newer one based on word embeddings.

1 Introduction

Motivation. People spend a significant amount of their time attending meetings. To benefit from recent technological advances, many companies are now using web-based meeting tools that can accommodate remote participants and allow video in addition to voice calls. While very useful, those tools typically do not offer extra features beyond screen sharing or instant messaging. In particular, they broadcast participant voices without leveraging the rich information conveyed in speech. Yet, the use of Automatic Speech Recognition (ASR) systems opens the gate to numerous text mining applications that can assist participants as the meeting unfolds, or once it is over.

Goals. Here, we focus on extracting keywords in real-time from speech transcriptions (ASR output) over the course of a virtual meeting. This task is very important, as current keywords provide a snapshot of the ongoing topics and can be used to improve productivity in a variety of ways: (1) on the fly retrieval of relevant internal and external resources (webpages, emails) based on the topics detected, (2) constant maintenance of a meeting summary to enable latecomers to quickly catch-up, and (3) smart indexing once the meeting is over.

Challenges. Processing multi-party meeting speech transcriptions is a difficult NLP task. First, spontaneous speech differs from traditional documents. In lieu of well-formed, self-contained *sentences*, the data consist of fragments of speech transcripts called *utterances*, which are often ill-formed, ungrammatical, and contain informal or filler words (e.g., "uh-huh"). Moreover, speakers dilute important information by frequently pausing, interrupting each other, and chit-chatting. Second, errors made by the ASR system inject some additional noise into the transcriptions.

Contributions.

1. We build on the k-core graph decomposition algorithm to assign scores to terms. As will be explained, our approach is particularly well suited to speech transcriptions as it is *fully unsupervised* and *robust to noise*.
2. To select the best terms, we propose a new *keyword quality function* and prove that it is *submodular*, which enables its near-optimal optimization under a budget constraint in a way fast enough to meet the real-time requirements.
3. We evaluate the performance of our method against that of numerous baselines on two standard, well-known datasets (AMI and ICSI), and reach state-of-the-art performance.
4. Finally, we release our code and data as publicly available[1], making our study *fully repro-*

[*]This research is supported in part by the OpenPaaS::NG project.

[1]`https://goo.gl/rIlDd6`

ducible. Furthermore, our system can be interactively tested online[2].

In the remainder of this paper, we introduce our system, describe our experiments, and report and interpret our results.

2 Proposed system

As shown in Figure 1, our system can be broken down into 4 modules. We describe them in what follows.

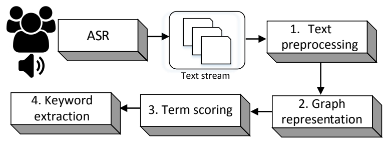

Figure 1: System architecture

2.1 Text preprocessing

T parameter. We receive as input a stream of text from the ASR tool, which is composed of utterances of duration 2.01s on average (std. dev. of 2.03). Starting from t=0 (beginning of the meeting), our system considers consecutive intervals I_i of fixed size T=60s. I_1 is made up of all utterances starting within $[0, T[$, I_2 covers $[T, 2T[$, etc. The number of words in each interval (before cleaning) is 200 on average (std. dev. of 75). T is a trade-off parameter: as it increases, more textual data become available for the interval, which usually yields better keywords. But on the other hand, additional lag is introduced. Note that we experimented with dynamic interval length based on speaker dominance periods, but found that while increasing complexity, it did not offer noticeable improvements.

Cleaning. At the end of each time period, we remove punctuation and standard stopwords from the associated utterances. We also filter out ASR-specific terms indicating inaudible sounds, pauses, and background noise, such as `{vocalsound}`. Finally, we perform tokenization and stemming.

2.2 Graph building

Then, from the pre-processed text for the interval, we generate an undirected, weighted graph of words $G(V, E)$ like in Mihalcea and Tarau (2004). Word co-occurrence networks are flexible, information-rich structures with many parameters (Tixier et al., 2016b). In the present study, the nodes V are unique terms (unigrams) in the text and two nodes are linked by an edge $e \in E$ if the two words they represent co-occur within a sliding window of fixed size $W = 3$ overspanning utterance boundaries (making our system robust to utterance segmentation errors). Furthermore, edge weights match co-occurrence counts. This step is $\mathcal{O}(|V|W)$ in time, which is very fast for the small graphs considered here ($|V| \approx |E| \approx 10$).

2.3 Term scoring

k-core. The k-core is one of the most fundamental constructs in network analysis. A maximal connected subgraph of G is said to be a k-core of G if each of its nodes has degree greater than or equal to k (Seidman, 1983). The core number of a node is the highest order of a k-core that contains this node.

k-core decomposition. We apply the generalized k-core algorithm of Batagelj and Zaveršnik (2011). Essentially, this algorithm deletes at each step the vertex of lowest degree (in the current subgraph) as well as all its incident edges, which decreases the degrees of the nodes in the neighborhood. Note that for a weighted graph, the degree of a vertex is the sum of the weights of its incident edges. As shown in Figure 2, the output is the k-core decomposition of G, that is, the set of all its cores from 1 (G as a whole) to k_{max} (its main core). The k-cores form a hierarchy of nested subgraphs whose cohesiveness and size respectively increase and decrease with k.

Application to keyword extraction. As we move upwards the k-core hierarchy of a graph of words, we expect to find more and more keywords. The underlying assumption is that in a word co-occurrence network, centrality (as measured by PageRank, for example) is not the best "keywordness" criterion, and that it is better instead to look for nodes that are not only central but that also form tightly knitted substructures with other nodes, that is, nodes that are part of *cohesive* subgraphs (Tixier et al., 2016a).

CoreRank. Finally, we assign to each node v in the graph the sum of the core numbers of its neighbors $\mathcal{N}(v)$:

$$cr(v) = \sum_{u \in \mathcal{N}(v)} core(u) \qquad (1)$$

We will refer to this scoring scheme as *CoreRank* in the remainder of this paper. Assigning scores at

[2] `http://83.212.204.91/conversations`

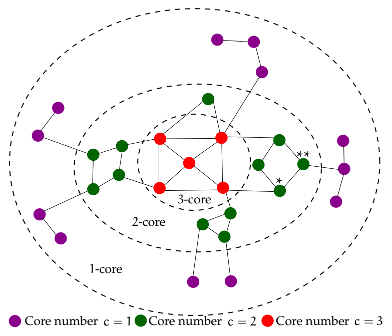

Figure 2: k-core decomposition of a graph and CoreRank (CR) scoring scheme. While nodes $\star$ and $\star\star$ have the same score (2) in terms of core numbers, node $\star$ has a greater CR score (7 vs 5), which accurately reflects its more central position in the graph.

the node level (rather than at the subgraph level) allows to better discriminate between vertices, which makes ranking and selection easier. Also, stabilizing scores across node neighborhoods increases robustness to noise, which is particularly desirable when dealing with noisy text like speech transcriptions.

Complexity. Computing the k-cores is very efficient: thanks to Batagelj and Zaveršnik (2011), it can be done in $\mathcal{O}(|V|+|E|\log|V|)$ time. Computing the CoreRank scores is also very affordable, as it is $\mathcal{O}(|E|)$ in time. For the small graphs considered here, these steps can therefore be performed very quickly, which suits well the real-time nature of our task.

2.4 Keyword extraction

Keyword quality function. Rather than using heuristics like in Tixier et al. (2016a) to select nodes from G (i.e., to extract tokens from the text), we frame the keyword identification problem as the maximization of a set function under a budget constraint. In particular, we define a *keyword quality function* f that not only measures the cumulative CoreRank score of a given set of terms S, but also the density of the subgraph they induce:

$$f(S) = \sum_{v \in S} cr(v) - \lambda h(S) \tag{2}$$

where λ is a trade-off parameter, and the set function h counts the number of edges that should be added to the subgraph induced by S to make it complete:

$$h(S) = \binom{|S|}{2} - |E(S)| \tag{3}$$

where $|S|$, resp. $|E(S)|$, denotes the number of vertices, resp. edges, in the subgraph induced by S. $h(S)$ is null when S is complete (i.e., of unit density), and increases as the density of the graph decreases. Recall that a complete graph is a graph where every two nodes are linked by an edge, and that a subgraph of $G(V, E)$ induced by a set of nodes $S \subseteq V$, has S as its vertices and all the edges from E for which both endpoints belong to S as its edges.

Interpretation. The first component of f measures the extent to which a set contains nodes with high CoreRank numbers, while its second term (h) provides an extra layer of cohesiveness requirements, by biasing the selection towards a set of nodes that together form a *dense* subgraph. To maximize f, we want to jointly maximize, resp. minimize, its first and second terms.

Optimization task. Finding the best subset of terms $S^* \subseteq V$ to serve as keywords can be seen as a combinatorial optimization task under a budget constraint:

$$S^* = \underset{S \subseteq V, \sum_{v \in S} c_v \leq B}{\arg\max} f(S) \tag{4}$$

where c_v is the unit cost of including term v as a keyword, and B is the budget, which we define as the number of keywords that should be returned. B can be expressed as a percentage of the total number of words in the interval, but here we consider it to be fixed.

Performance guarantees. As we prove in the extended version of this paper, our keyword quality function f is submodular, enabling Equation 4 (NP-complete) to be solved by a simple greedy algorithm with $(1-{}^1\!/_e) \approx 0.63$ approximation guarantees (Nemhauser et al., 1978). Note that to benefit from these guarantees, f should also be monotone, which does not apply in our case. However, we invoke the fact that if $|S| \ll |V|$ (which holds here), the monotonicity constraint can be relieved (Lin et al., 2009; Krause, 2008).

3 Experimental Setup

3.1 Datasets

We used two datasets well-known in the field of meeting speech processing: the AMI corpus[3] (McCowan et al., 2005) and the ICSI corpus[4] (Janin et al., 2003). These datasets contain respectively 137 and 57 meetings lasting from 10 to 70 minutes

[3] `http://groups.inf.ed.ac.uk/ami/corpus/`
[4] `http://www1.icsi.berkeley.edu/Speech/mr/`

(2,400 to 19,000 words) and involving between 2 and 6 participants whose conversations were automatically converted to text with a word error rate approaching 37%. Each meeting comes with gold standard in the form of human-written abstractive and extractive summaries. The extractive summaries were put together by selecting the best utterances from the transcripts. In some cases, multiple summaries are available for the same meeting.

3.2 Baselines

We evaluated the performance of our system against that of 5 baselines and an Oracle, which are presented next.

First, to better interpret our results and enable easy cross-comparison with other studies, we included two standard, basic baselines: (1) selecting words at random from the processed text (without replacement), and (2) selecting the most frequent words from the processed text. Within our graph-based submodular framework, we also considered the replacement of CoreRank scores with (3) weighted degree centrality (sum of the weights of the incident edges), (4) PageRank scores (Mihalcea and Tarau, 2004), and (5) RAKE scores: $^{deg(v)}/_{freq(v)}$, where $deg(v)$ is the weighted degree of term v in the graph and $freq(v)$ its frequency in the text (Rose et al., 2010). Finally, we used as an Oracle the (6) most frequent words from the part of the extractive summary corresponding to the time interval considered. Of course, we used the same budget for all baselines, the Oracle, and our system.

3.3 Evaluation methodology

We compared all systems under two settings.

Scenario 1. Using the traditional vector-space model, we computed the cosine similarity between the sum of the one-hot vectors of the keywords returned by a given method for a particular time interval, and the sum of the one-hot vectors of the words in the part of the extractive summary corresponding to the same interval. Results were averaged across summaries (when multiple ones were available), and finally across time intervals to compute the overall performance of the method (macro-averaging). For the random baseline, results were first averaged over 10 runs, to reduce variance. In this scenario, the method whose keywords most closely match the gold standard receives the highest score. Note that using TF-IDF weighting (rather than integer entries) did not change the rankings.

Scenario 2. For the sake of completeness, we also wanted to evaluate performance against the *abstractive* summaries. However, since the sentences in those summaries do not come from the transcripts but were freely written by annotators, they are not time-stamped and thus cannot be linked to any particular interval. Consequently, to allow comparison, we concatenated the keywords extracted by a given method and for a given meeting from all intervals, thus obtaining a concise keyword-based summary of the full meeting. To compute the similarity with the abstractive summaries, we then used ROUGE-1 (Lin, 2004) and the Word Mover's Distance (WMD) (Kusner et al., 2015). ROUGE-1 computes similarity based on unigram overlap, while the WMD takes into account semantic similarity between terms, and is therefore more robust to the fact that the abstractive summaries contain words that were never actually spoken. Very briefly, the WMD is the minimum cumulative Euclidean distance needed for all words in the first summary to travel (in an embedding space) to the second summary. As our embeddings, we used publicly available[5] 300-dimensional vectors learned by Mikolov et al. (2013) from a 100B-word corpus (Google News). Note that since the WMD is a distance, the best performing methods are associated in that case with the *lowest* scores (for ROUGE, which is a measure of similarity, it is the opposite).

4 Results

Tables 1 and 2 display the results for the first and second scenarios, respectively. In both cases, and on both the AMI and ICSI corpora, CoreRank outperforms the baselines, sometimes by a wide margin. Overall, the Oracle reaches best performance, which was expected since it has direct access to the gold standard. Nevertheless, it highlights the fact that there is still much room for improvement. However, it is worth noting that on the AMI dataset, under the second scenario, CoreRank outperforms even the Oracle.

Impact of the budget. Figures 3 and 4 report the results in scenario 1, respectively for the AMI and ICSI datasets, for an increasing number of extracted keywords. The curves of the Oracle, Random and RAKE baselines were omitted for

[5] `https://code.google.com/archive/p/word2vec/`

readability purposes. On both datasets, as the number of extracted keywords increases, we observe that the performance of all methods also increases. However, the rankings remain stable.

Impact of h. Under the first setting and on the AMI corpus, we finally investigated how the density term (h) of our submodular function f was influencing the performance of the graph-based systems. As shown in Table 3, h proved beneficial, even though the improvements were marginal. The only exception was RAKE, for which best performance was achieved for $\lambda = 0$ (no density term). Note that the trade-off parameter λ was optimized for each method on a small development set consisting of 60 time intervals randomly drawn (without replacement) from the AMI corpus. We searched the $[0, 3]$ line, with uniform steps of size 10^{-3}.

5 Related work

To the best of our knowledge, this study is the first to investigate the extraction of keywords from meeting speech transcriptions in *real-time*. However, previous work did focus on *offline* meeting summarization. For instance, Lin et al. (2009) used a sentence semantic graph and a different submodular objective function. Habibi and Popescu-Belis (2013) used LDA and submodularity to select keywords covering as many topics as possible. Here, we assume that at most one topic can be discussed within each of our short time intervals. Closely related to our work is also that of Meladianos et al. (2015), who detected sub-events in real-time from the Twitter stream by stacking graphs of terms built from full tweets (without sliding window) and studying the evolution of core numbers over time in the overall graph. In our case, however, utterances are not self-contained pieces of information, and we don't receive them at a rate that is high enough to enable any kind of temporal analysis.

6 Conclusion

We presented a novel approach for real-time keyword extraction from ASR output, based on the core decomposition of networks and submodularity. Results show the superiority of our method over several baselines.

[6] http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ranksums.html

Method \ Dataset	AMI	ICSI
Oracle	0.849	0.758
CoreRank	**0.474***	**0.259***
PageRank	0.469	0.250
Degree	0.470*	0.245
Frequency	0.460	0.231
RAKE	0.384	0.196
Random	0.365	0.190

Table 1: Results for scenario 1 (real-time, cosine similarity). * indicates statistical significance[6] at $p < 0.05$ against the *Frequency* baseline of the same column.

Method \ Dataset	AMI		ICSI	
	ROUGE	**WMD**	**ROUGE**	**WMD**
Oracle	22.7	1.582	13.6	1.052
CoreRank	**23.7**	**1.653**	**13.4**	**1.699**
PageRank	21.9	1.657	13.3	1.701
Degree	21.3	1.657	13.0	1.712
Frequency	21.4	1.661	12.1	1.709
RAKE	19.5	1.724	10.8	1.705
Random	16.1	1.761	7.7	1.772

Table 2: Results for scenario 2 (keyword-based summary of the entire meeting). With ROUGE, greater is better, while with WMD, lower is better.

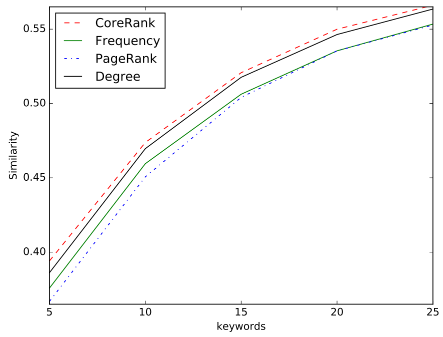

Figure 3: Performance in scenario 1 (cosine similarity) for a varying number of extracted keywords, on the AMI corpus.

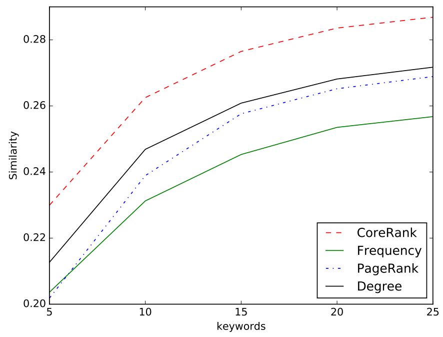

Figure 4: Performance in scenario 1 (cosine similarity) for a varying number of extracted keywords, on the ICSI corpus.

method	$\lambda = 0$	optimal λ
CoreRank	.470	.474
PageRank	.466	.469
Degree	.467	.470

Table 3: Performance in scenario 1 and on the AMI corpus, with and without the density-based term of f.

References

Vladimir Batagelj and Matjăz Zaveršnik. 2011. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classification*, 5(2):129–145.

Maryam Habibi and Andrei Popescu-Belis. 2013. Diverse Keyword Extraction from Conversations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 651–657.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. 2003. The icsi meeting corpus. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–364.

Andreas Krause. 2008. *Optimizing Sensing: Theory and Applications*. ProQuest.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Weinberger Kilian Q. 2015. From Word Embeddings To Document Distances. In *Proceedings of the 32th International Conference on Machine Learning*, pages 957–966.

Hui Lin and Jeff Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 510–520.

Hui Lin, Jeff Bilmes, and Shasha Xie. 2009. Graph-based Submodular Selection for Extractive Summarization. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 381–386.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics - Workshop: Text Summarization Branches Out*, pages 74–81.

Iain McCowan, Jean Carletta, W Kraaij, S Ashby, S Bourban, M Flynn, M Guillemot, T Hain, J Kadlec, V Karaiskos, et al. 2005. The ami meeting corpus. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, volume 88.

Polykarpos Meladianos, Giannis Nikolentzos, François Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2015. Degeneracy-based Real-Time Sub-Event Detection in Twitter Stream. In *Proceedings of the 9th AAAI Conference on Web and Social Media*, pages 248–257.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining*, pages 3–20.

Stephen Seidman. 1983. Network Structure and Minimum Degree. *Social networks*, 5(3):269–287.

Antoine J-P. Tixier, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. 2016a. A Graph Degeneracy-based Approach to Keyword Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1860–1870.

Antoine J-P. Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. 2016b. Gowvis: a web application for graph-of-words-based text visualization and summarization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics System Demonstrations*, pages 151–156.

7 Appendix

Theorem 1. *Function f is submodular.*

Proof. We first show that the set function h is supermodular. Let $A \subseteq B \subseteq V \backslash v$. When adding vertex v to A, $h(A \cup v) - h(A)$ is equal to the number of nodes in A that are not connected to v (see Figure 5). Likewise, $h(B \cup v) - h(B)$ corresponds to the number of nodes in B that are not linked to v. Since $A \subseteq B$, the number of nodes in B that are not linked to v is at least equal to the number of nodes in A that are not linked to v. Hence, $h(B \cup v) - h(B) \geq h(A \cup v) - h(A)$, which is equivalent to saying that h is supermodular. Thus, for any $\lambda \geq 0$, $-\lambda h(S)$ is submodular. Since $\sum_{v \in S} cr(v)$ is also submodular (Lin and Bilmes, 2011), and the sum of two submodular functions is submodular, f is submodular. $\square$

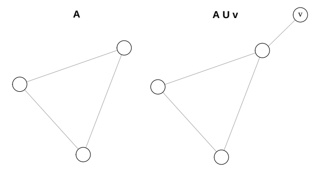

Figure 5: $h(A) = 0$ and $h(A \cup v) = 2$ (2 edges are missing to get a complete graph) $\Rightarrow h(A \cup v) - h(A) = 2$. The is also the number of nodes of $A \cup v$ not linked to v