ClusterRank: A Graph Based Method for Meeting Summarization

Conference Paper · January 2009

Source: DBLP

CITATIONS
36

READS
316

4 authors, including:

Korbinian Riedhammer
Technische Hochschule Nürnberg Georg Simon Ohm
41 PUBLICATIONS **692** CITATIONS
SEE PROFILE

Dilek Hakkani-Tur
Microsoft
278 PUBLICATIONS **5,749** CITATIONS
SEE PROFILE

IDIAP RESEARCH REPORT

CLUSTERRANK: A GRAPH BASED METHOD FOR MEETING SUMMARIZATION

Nikhil Garg Benoit Favre Korbinian Reidhammer
Dilek Hakkani Tür

Idiap-RR-09-2009

JUNE 2009

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

ClusterRank: A Graph Based Method for Meeting Summarization

Nikhil Garg[1]*, Benoit Favre*[2]*, Korbinian Reidhammer*[2]*, Dilek Hakkani-Tür*[2]

[1]Ecole Polytechnique Fédérale de Lausanne, Switzerland
[2]International Computer Science Institute, Berkeley, USA
nikhil.garg@epfl.ch, {favre,koried,dilek}@icsi.berkeley.edu

Abstract

This paper presents an unsupervised, graph based approach for extractive summarization of meetings. Graph based methods such as TextRank have been used for sentence extraction from news articles. These methods model text as a graph with sentences as nodes and edges based on word overlap. A sentence node is then ranked according to its similarity with other nodes. The spontaneous speech in meetings leads to incomplete, ill-formed sentences with high redundancy and calls for additional measures to extract relevant sentences. We propose an extension of the TextRank algorithm that clusters the meeting utterances and uses these clusters to construct the graph. We evaluate this method on the AMI meeting corpus and show a significant improvement over TextRank and other baseline methods.
Index Terms: Summarization, Page Rank

1. Introduction

Text summarization has gained significant interest for news articles and web pages. Given a text, the summarization task consists in extracting or forming sentences to be included in the summary such that they cover important information with minimal redundancy, while satisfying a length constraint. For meeting summarization, transcripts of participants' dialogs may be generated using automatic speech recognition (ASR). However, the nature of spontaneous speech presents additional challenges such as incomplete or ill-formed sentences, noise in the form of random "chit-chat", and high redundancy which should be taken into account while forming a summary. For instance, consider the dialog in Figure 1. In this discourse, Person A

[1] A: Due to severity of this problem
[2] B: Which problem?
[3] A: Global warming.
[4] B: Oh. Yeah, right
[5] A: This severe problem has made us look into other sources of energy

Figure 1: Dialog between two persons: A and B

might have initially intended to say that 'The severity of Global Warming has made us look into other sources of energy'. The information in this conversation is split across five sentences or *utterances*. As discussed later, this is one of the motivations for *clustering* the utterances together that address the same subject. Furthermore, a summary should avoid including redundant sentences that do not convey any new information.

A summary can either be extractive i.e. consisting of sentences chosen from the given text, or abstractive where new sentences might be synthesized or existing ones rephrased by the system. Most of the previous work has focused on extractive summarization of structured text like news articles. While these methods can be applied in meetings as well, taking additional measures to cope with high noise and redundancy could improve the performance of the summarization algorithms. In this work, we present an extension of the TextRank algorithm [1], ClusterRank, that segments the transcript into clusters and uses the clusters to construct a cluster graph of the transcripts. We evaluate our approach against TextRank and other algorithms used for meeting summarization.

Section 2 describes related work on summarization. Section 3 and 4 gives a brief overview of the TextRank algorithm and a detailed description of our approach. Dataset and Results are presented in sections 5 and 6, respectively. We conclude in section 7 and discuss some future directions of research.

2. Related Work

Several methods have been proposed for ranking sentences based on some relevance metric. In supervised approaches [2], a classifier is trained on sentence features such as key words, sentence length, and position, and given a new text, a relevance score for each sentence is calculated. [3] uses the structural features of broadcast news such as position, length and speaker of the segment to rank different segments. Maximal Marginal Relevance (MMR) [4] is an unsupervised query based approach where a term vector is created for each sentence. In the absence of a query, a centroid of all the term vectors is used. Sentences are chosen iteratively to be included in the summary such that they are maximally similar to the query and have minimal similarity to the already chosen sentences. In meetings, such an approach tends to promote non-content words such as "okay" and "sort of". [5, 6] tackle this issue by using *keyphrases*, extracted from the text, to construct the centroid. [7] augmented the textual features with a set of prosodic features such as F0, energy and duration of utterances, and found that much effort is needed to find the right set of features for sentence extraction. [8] models the text as a bipartite graph having a node for every sentence and word, and a weighted undirected edge from a sentence node to a word node if the word appears in the sentence. The author argues that sentences with high *saliency* scores should have high saliency words in them and words that appear in many high saliency sentences should have high saliency scores. This mutual reinforcement principle is used to find high saliency keywords and sentences in the text. Other graph based methods include eigenvector centrality approaches that have been applied in TextRank [1] and LexRank [9]. Our approach is an extension of the TextRank algorithm as described in the next section.

3. TextRank

TextRank [1] is a graph-based sentence extraction algorithm where each sentence is represented by a node in the graph. An undirected edge between two sentences is created based on lexical similarity. Specifically, if a sentence S_i is represented as a set of words: $S_i = w_1^i, w_2^i, ...w_{|S_i|}^i$, then the similarity between two sentences S_i and S_j is defined as:

$$Sim(S_i, S_j) = \frac{|\{w_k : w_k \in S_i \wedge w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)}$$

An edge based on similarity can be seen as a process of "recommendation": a sentence that addresses certain concepts, gives the reader a "recommendation" to refer to other sentences that address the same concepts. The underlying assumption for calculating relevance is that the sentences which are similar to a large number of other important sentences are "central". TextRank uses PageRank [10] to calculate a relevance score for each sentence based on the relevance score of its similar sentences. Top ranked sentences are selected for the summary such that their total length satisfies the summary length constraint.

As reported in [1], TextRank works quite well for structured text like news articles where each sentence has some meaningful information and redundancy is low. However, in spontaneous speech, the utterances are typically not well-formed. Participants often interrupt each other during the conversation leading to information spread across multiple utterances. In the conversation shown in Figure 1, even though it might be important, utterance 5 would get a lower score compared to if it had the keywords 'global' and 'warming' in it, as then it would have shared more keywords with the rest of the transcript. Due to this feature of spontaneous speech, it is useful to consider the above five utterances together as a *cluster* rather than individual utterances. Since a cluster is likely to contain more keywords, a good scoring method would probably give a high rank to the cluster compared to if they were considered as individual utterances. We use text segmentation to form utterance clusters, and create cluster nodes rather than utterance nodes in the graph.

A meeting might also have "off-topic" sections unrelated to the general theme of the meeting. For instance, in a meeting about 'effects of climate change', there might be a brief exchange about 'what is for lunch today'. We would like to avoid including such utterances in a summary. In an utterance based graph, the utterances in an off-topic segment would link to each other and as a result might be assigned a high score even if they share very few keywords with the rest of the transcript. If the entire off-topic conversation is merged into a single cluster, that cluster is unlikely to be linked to other clusters with high weight edges due to lack of common keywords. Thus, the off-topic cluster gets down-weighted.

Redundancy is a major problem in meetings where the same information is often repeated by participants. We employ a simple solution to this problem by representing the present summary as a bag of words. A new sentence is picked only if its similarity with this bag is below a certain threshold.

4. ClusterRank

As a preprocessing step, we remove the stopwords and apply Porter's stemming algorithm [11]. The following subsections describe the five steps of our algorithm.

4.1. Clustering

The aim of clustering or text-segmentation is to segment the transcript such that each section has utterances about the same subject. TextTiling [12] is a domain-independent algorithm for multi-paragraph segmentation of text. Text is subdivided into pseudo-sentences or token sequences of predefined length. A potential segment break is then evaluated based on the lexical similarity of adjacent blocks of token sequences. LCseg [13] computes lexical chains consisting of term repetitions and proposes that major topic shifts are likely to occur where "strong" term repetitions start and end. [14] presents a bayesian approach to unsupervised topic segmentation where a multinomial language model is associated with every topic segment. Since our aim is to demonstrate the usefulness of segmentation, we chose a simple approach similar to TextTiling [12] by starting with each sentence as a separate cluster and merging adjacent clusters if they are similar above a certain threshold. We define the weight of a word w in a cluster X as:

$$W_X(w) = freq_X(w) \times IDF(w)$$

where $freq_X(w)$ denotes the number of sentences in cluster X that have the word w and $IDF(w)$ is the inverse document frequency of the word w calculated as:

$$IDF(w) = log\left(\frac{\text{\# of sentences}}{\text{\# of sentences with word } w}\right)$$

We compute the similarity between two clusters using the cosine similarity of words they contain.

$$sim(X,Y) = \frac{\sum_{\{w:w\in X \wedge w\in Y\}} W_X(w) * W_Y(w)}{\sqrt{\sum_{w\in X} W_X^2(w)}\sqrt{\sum_{w\in Y} W_Y^2(w)}}$$

```
Initialize:  window_above = 1, window_below = 1

while max(window_above, window_below)<=window_threshold:

    for each possible merge_point:
        block_above = block of clusters in the range
                [merge_point-window_above .. merge_point]
        block_below = block of clusters in the range
                [merge_point+1 .. merge_point+window_below]
        sim = cosine_similarity(block_above, block_below)
    choose a merge_point for which sim is maximum

    if sim >= similarity_threshold
        merge_clusters(block_above, block_below)
    else
        update_windows(window_above, window_below)
```

Figure 2: Clustering algorithm. The update_windows() function generates a combination of (window_above, window_below) in the following order: {(1,1),(1,2),(2,1),(2,2), (1,3), (3,1), (2,3), (3,2), (3,3)...(max_window,max_window)}.

Figure 2 gives the complete clustering algorithm. The algorithm works by calculating the similarity between all pairs of adjacent clusters, selecting the pair with the highest similarity and merging it into a single cluster. The process is repeated as long as an eligible pair can be found. As illustrated in Figure 3, merging only two clusters at a time might be too restrictive. Due to incompleteness of sentences, adjacent sentences might not have enough common words even though they are about the same subject. To get a more effective clustering, we choose a *merge_point* and windows of size $window\_above$ and $window\_below$ where $1 \leq max\{window\_above, window\_below\} \leq window\_threshold$. We optimize these parameters using a held-out set. For our dataset, the parameters were $similarity\_threshold = 0.4$ and $window\_threshold = 3$. We observed that the segmentation of sentences tends to stabilize after a certain $window\_threshold$ value as the clusters become big in size and the similarity between two clusters no longer satisfies the $similarity\_threshold$. There is an initial increase in the summarization performance due to clustering. As we increase $window\_threshold$ beyond 4, the probability of wrong segmentation increases, resulting in a slight decline in performance.

4.2. Graph Construction

A cluster X can be treated as a bag of words with the weight of each word as $W_X(w)$. Each cluster is represented as a node in the graph. A directed edge from cluster X to cluster Y is weighted by their normalized similarity value. More precisely,

$$edge(X,Y) = \frac{sim(X,Y)}{\sum_Z sim(X,Z)}$$

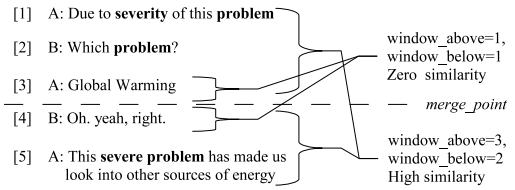

Figure 3: Effect of window sizes on clustering

An important difference to note here, compared to the TextRank algorithm, is that instead of simply counting the number of common words between two nodes, we take the cosine similarity of the words where each word w is weighted by $freq(w) * IDF(w)$. This gives a higher similarity score if the two nodes share less-common keywords. We confirmed experimentally that cosine similarity indeed gives slightly better performance for both TextRank and ClusterRank, and show results with cosine metric in both algorithms.

4.3. PageRank

The obtained graph can be represented as a stochastic matrix G in which an entry (i, j) denotes the edge weight from node j to node i. Further, to make the matrix primitive and irreducible, we construct the final matrix M as:

$$M = d * G + (1 - d) * (1/|nodes|) * I^{(1)}$$

where d is the damping factor taken as 0.85, as suggested in [10]. $I^{(1)}$ is a square matrix with all entries as one. Next, we use the PageRank algorithm [10] to compute an "importance" score for each cluster.

4.4. Sentence Scoring

To go from cluster scores to sentence scores, a reasonable approach would be to simply compute the centroid of the sentences within the cluster and rank the sentences according to their similarity with the centroid. However, the sentences within a cluster might also contain random "chit-chat" words unrelated to the rest of the meeting. To avoid their influence, we construct the centroid by taking into account only the words shared with other clusters. Also, since the PageRank score of the cluster depends on the word overlap with other clusters, this is a more reliable estimate of the importance of a sentence within the cluster. For a cluster X, the weight of a word w in the centroid C_X is calculated as:

$$C_X(w) = \sum_{Y:w \in Y, Y \neq X} PR(Y) * M(X, Y) * IDF(w)$$

where $PR(X)$ denotes the page rank score of cluster X. This gives a higher weight to words common with high PageRank clusters. The score of a sentence s in a cluster X is taken as:

$$score(s) = sim(s, C_X) * PR(X)$$

where $sim(s, C_X)$ refers to cosine similarity, calculated as:

$$sim(s, C_X) = \frac{\sum_{\{w:w \in s \wedge w \in C_X\}} W_s(w) * C_X(w)}{\sqrt{\sum_{w \in s} W_s^2(w)} \sqrt{\sum_{w \in C_X} C_X^2(w)}}$$

$W_s(w)$ is taken as $IDF(w)$. $score(s)$ is calculated such that a sentence in a high PageRank cluster will get down-weighted if it has a very low similarity with other clusters. This helps filter out noise from the summary.

4.5. Greedy Selection

Once we have a score for each sentence, we start from the highest scoring sentence and include sentences in the summary until the length constraint is satisfied. At any iteration step, the summary already constructed, sum, is represented as a bag of words with the weight of a word w as:

$$W_{sum}(w) = |\{S_i : w \in S_i \wedge S_i \in sum\}| * IDF(w)$$

To minimize redundancy, for a new sentence S, we calculate a ratio of the words in S that are already present in the summary:

$$r = \frac{\sum_{w \in S} W_{sum}(w)}{\sum_{w \in S} IDF(w)}$$

If r is below a certain threshold β and it satisfies the length constraint, we include S in the summary.

In a variation of this algorithm, we first normalize the sentence scores by their length and pick the sentences in the decreasing order of '$score(s)/log(1 + length(s))$' value. This gives preference to shorter sentences which have high scores. Selecting shorter sentences helps create more space in the summary for other topics thus leading to an increase in recall. We call this variation "ClusterRank with normalization" and also include its results along with the basic ClusterRank algorithm.

5. Data

The AMI meeting corpus [15] is a collection of 100 hours of meeting data that includes speech audio, transcripts, and human summaries. Each meeting has participants talking for about 35 minutes on a given topic and the transcripts are about 3000-7000 words. Our experiments are based on the 137 meetings that have abstractive human summaries.

6. Evaluation and Results

For each meeting, we generate an extractive summary satisfying a length constraint specified in terms of a percentage of the total number of words in the transcript. We compare the performance to the following algorithms:

- TextRank with cosine similarity metric
- Centroid based MMR algorithm
- Longest sentence baseline that iteratively forms the summary by choosing the longest sentence at each step.

ROUGE [16] is used to evaluate the performance for all the systems. ROUGE scores are based on the number of overlapping units such as n-grams, between the system generated summary and the ideal summaries created by humans. The human summaries provided in the corpus have a length of about 6% words. Table 1 shows the ROUGE-1 (unigram overlap) and ROUGE-2 (bigram overlap) precision, recall, and F-measure for different algorithms for a summary length constraint of 6% words. As shown in the table, ClusterRank achieves a significant performance gain over the basic TextRank model. Normalizing sentence scores by their lengths gives a further improvement in recall. However, as observed in our summaries, selecting shorter length sentences may decrease the readability of the summary, which is in general a problem for extractive summarization of meetings. Human evaluation can provide more insights, and this remains as an issue for future research. The performance comparison follows a similar trend for both human transcripts as well as ASR output. The performance difference of different algorithms remains consistent across different summary lengths as shown in Figure 4.

	Algorithm	Rouge-1			Rouge-2		
		Precision	Recall	F-measure	Precision	Recall	F-measure
Human Transcripts	Longest Sentence Baseline	0.257	0.146	0.182	0.037	0.021	0.026
	Centroid based MMR	0.258	0.147	0.182	0.042	0.023	0.029
	TextRank with cosine similarity	0.327	0.212	0.250	0.062	0.040	0.047
	ClusterRank	**0.361***	0.232*	0.275*	**0.069**	0.043	0.051
	ClusterRank with sentence normalization	0.351*	**0.262***	**0.291***	0.067	**0.049***	**0.055***
ASR Output	TextRank with cosine similarity	0.302	0.219	0.246	0.055	0.039	0.044
	ClusterRank	**0.337***	0.240*	0.272*	**0.059**	0.041	0.047
	ClusterRank with normalization	0.322	**0.262***	**0.280***	0.056	**0.044**	**0.048**

Table 1: Performance of different algorithms for a 6% summary length. The numbers in bold denote the best performing algorithm for a given measure. * denotes a significant improvement over the TextRank algorithm according to t-test at 95% confidence interval.

The clustering approach tends to suppress topics that are unrelated to the general topic of the meeting. If one does not want to suppress these topics, it might be advisable to avoid clustering and construct a sentence graph instead. The AMI corpus includes manual topic annotations for meetings, but the topics there are very broad which leads to very few clusters. The PageRank scores will not be meaningful in this case and a better sentence scoring could be achieved by only considering sentence similarity within the cluster. An extreme case would be just a single cluster and here, the approach could be a centroid based MMR algorithm. On the other extreme, the graph constructed by ClusterRank would be very similar to TextRank if there are as many clusters as sentences.

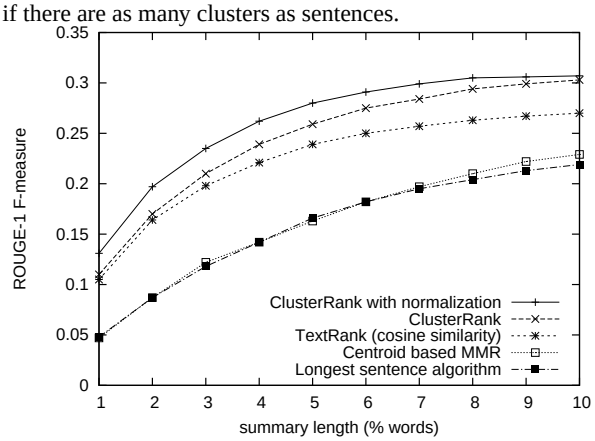

Figure 4: Performance variation with summary length

7. Conclusion and Future Work

We have presented an unsupervised graph based algorithm for extractive summarization of meeting transcripts. Our approach extends a text summarization algorithm, TextRank, to include measures for high noise and redundancy found in meetings. For future work, we are investigating more sophisticated techniques for text-segmentation that look beyond the simple word overlap. Another enhancement would be to integrate a user query into the system by giving higher weights to "topics" present in the query. This could be done along the lines of topic sensitive page-rank [17] where the edge weights in the link graph are adjusted according to the query topic.

Acknowledgments: This work was funded by Swiss National Science Foundation (SNSF) through National Center for Competence in Research (NCCR) on Interactive Multimodal Information Management (IM2).

8. References

[1] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Texts," in *Proceedings of EMNLP 2004*. Association for Computational Linguistics, 2004, pp. 404–411.

[2] J. Kupiec, J. Pedersen, and F. Chen, "A Trainable Document Summarizer," in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM New York, USA, 1995, pp. 68–73.

[3] S. R. Maskey and J. Hirschberg, "Automatic Summarization of Broadcast News Using Structural Features," in *8th European Conference on Speech Communication and Technology*. ISCA, 2003.

[4] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell, "Summarizing text documents: sentence selection and evaluation metrics," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM Press New York, USA, 1999, pp. 121–128.

[5] K. Riedhammer, B. Favre, and D. Hakkani-Tür, "A Keyphrase Based Approach to Interactive Meeting Summarization," in *Spoken Languge Technologies (SLT), Goa (India)*, 2008.

[6] S. Xie and Y. Liu, "Using corpus and knowledge-based similarity measure in Maximum Marginal Relevance for meeting summarization," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*, 2008, pp. 4985–4988.

[7] G. Murray, S. Renals, and J. Carletta, "Extractive Summarization of Meeting Recordings," in *Ninth European Conference on Speech Communication and Technology*. ISCA, 2005.

[8] H. Zha, "Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM New York, USA, 2002, pp. 113–120.

[9] G. Erkan and D. R. Radev, "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization," *Journal of Artificial Intelligence Research*, vol. 22, no. 2004, pp. 457–479, 2004.

[10] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[11] M. Porter, "The Porter Stemming Algorithm," *Accessible at http://www.tartarus.org/martin/PorterStemmer*, 1980.

[12] M. A. Hearst, "Multi-Paragraph Segmentation of Expository Text," in *Annual Meeting-Association for Computational Linguistics*, vol. 32. Association for Computational Linguistics, 1994.

[13] M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing, "Discourse segmentation of multi-party conversation," in *Annual Meeting-Association for Computational Linguistics*, vol. 1. Association for Computational Linguistics, 2003.

[14] J. Eisenstein and R. Barzilay, "Bayesian Unsupervised Topic Segmentation," in *Proceedings of EMNLP 2008*. Association for Computational Linguistics, 2008.

[15] I. McCowan, J. Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos *et al.*, "The AMI Meeting Corpus," in *Proceedings of Measuring Behavior*, 2005.

[16] C. Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, 2004, pp. 25–26.

[17] T. H. Haveliwala, "Topic-sensitive PageRank," in *Proceedings of the 11th International World Wide Web Conference*, 2002.