bigIR at TREC 2019: Graph-based Analysis for News Background Linking

Marwa Essam and Tamer Elsayed
{me1709534,telsayed}@qu.edu.qa
Computer Science and Engineering Department
Qatar University
Doha, Qatar

ABSTRACT

Nowadays, it is very rare to find an online news article that is self-contained with everything a reader would want to know about the article's story. Therefore, it became vital for any article to contain links to other articles or resources that provide the background and contextual knowledge required to conceptualize the article's story. However, finding useful background and contextual links can be a challenging problem. In this paper, we address this problem in the context of the participation of the bigIR team at Qatar University in the news background linking task of the TREC 2019 news track. Our methods mainly relied on a graph-based analysis of the query-article's text to extract its most representative and influential keywords, and then use these keywords as a search query to retrieve the article's background links from a collection of news articles. All of our submitted runs outperformed the TREC hypothetical run that achieved a median effectiveness over all queries. Moreover, our best submitted run was ranked second among 28 runs submitted to the task, indicating the potential effectiveness of our approach.

1 INTRODUCTION

In the last decade, online news reporting services have played a vital role in changing the way people receive and share news. Often though, an author of any news article assumes that readers have a certain background knowledge on the article's topic or story. This is, of course, not the case for many readers. Therefore, to help readers conceptualize a news article, it is necessary to add links in each article to other articles or resources that provide the desired background knowledge [8]. Links in articles are typically added to point to other articles that are written by the same author, or ones that are most-frequently viewed by the readers. However, adding useful background links to articles can be a challenging problem.

Motivated to find solutions to this problem, a news background linking task was recently introduced as a new challenge in the news track in TREC 2018 [8]. A number of teams participated in this challenge in 2018 proposing different methods to solve the problem. Most of the proposed methods relied mainly on an ad-hoc search approach to retrieve the background links for query articles. In this approach, a search query/queries are constructed from an input article, and these queries are issued against an index created for the news collection to retrieve the background links for the input article. Upon the retrieval of an initial set of background links, some of the earlier proposed methods further adopted other approaches to extend their initial results or even to diversify the background links presented to the reader [2–4, 10].

In this paper, we demonstrate the participation of our bigIR team at Qatar University in the background linking task in TREC 2019. Our approach relied also on an ad-hoc search approach to find the required background links; nonetheless, we base the construction of the search query mainly on a graph-based analysis of the query article's text [5]. Precisely, we adopted graph construction and decomposition methods to extract a set of keywords from the query article, and used a weighted representation of those keywords as our search query. Our motivation for using graph decomposition methods for keyword extraction is that it catches important information in the text like the order, co-occurrence, and context relations between the different terms, allowing the most influential and representative keywords to stand out. We submitted four runs to this task using four different methods; yet, all our methods relied on the same graph-based analysis idea.

The rest of this paper is organized as follows. Section 2 describes, in detail, our proposed approach to solve the problem. Section 3 presents the official runs setup and results as reported back by TREC. Finally, section 4 concludes the paper and outlines potential future work.

2 APPROACH

We map the original news background linking problem to the following problem:

Given a collection of news articles D and a query article A, construct a search query Q to retrieve background articles of A from D.

To construct the search query Q, we extract keywords from the query article A that best indicates its relevant topics, and concatenate these keywords in one query. Specifically, the required keywords are obtained using a graph-based analysis of the query article A. Graph-based analysis of text allows to capture the dependency between different terms. Dependency relations may exist between terms in a document (e.g., grammatical dependency), and thus can help quantify the importance of those terms by relative weights. Consequently, those weights may help in choosing keywords that better represent the document. Recently, graph representation of text was explored to deal with term dependency in various Information Retrieval (IR) tasks, and was proved effective [6, 7, 9].

In this section, we first present the core architecture of our background links retrieval system, then we discuss each component in the system in detail.

2.1 System Design

Figure 1 depicts the high-level architecture of our proposed system. Initially, all the articles in the news collection are indexed. To index an article, its metadata is extracted first. This may include the article's title, URL, author's name and publishing date. For now, we

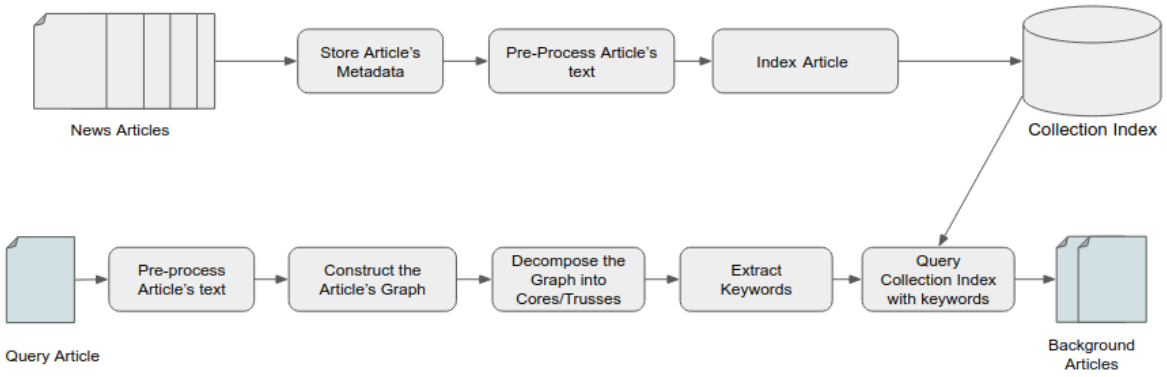

Figure 1: A high level overview of the background links retrieval system

used this metadata to filter out duplicate articles or articles that represent personal opinions or views. Nonetheless, this metadata can further be used in the re-ranking process of the retrieved results to support the diversification and the freshness of the final articles presented to the reader. Each article then goes to a preprocessing phase where a token stream is generated and fed to the indexer.

Having all articles in the collection indexed, we proceed to process query articles. For each query article, a co-occurrence graph is constructed from the article's title and body concatenated together. The article's graph is then decomposed into subgraphs that help assign weights to the terms, and the keyword set of the required query text Q is extracted accordingly. Q is finally used to retrieve the required background links from the collection index.

2.2 Graph Construction

After preprocessing the query article A into a stream of terms, we construct a *co-occurrence graph* G for the article. The underlying assumption of creating such graph is that terms co-occurring within a relatively small window of text have some kind of *semantic relatedness* regardless of their roles in a sentence, and that this relationship influences the importance of each single term within the article. Each node in G represents a single (unigram) term, and each edge connects two terms that co-occur (within a sliding window) at least once in the article. Edges are weighted by the co-occurrence frequency of the term pair. Figure 2 shows the graph constructed for an example short article. We opted to construct an *undirected* graph as recommended in [5], since using directed graphs was not proved to achieve significant impact on keyword extraction.

2.3 Graph Decomposition

After constructing the graph, weights of nodes (representing terms) can be computed using different measures. In this work, we select keywords based on graph decomposition methods [9]. The idea behind those methods is to decompose the co-occurrence graph into a hierarchy of nested *subgraphs* using graph degeneracy methods. Each *subgraph* contains a cohesive subset of nodes from the initial graph. Going down this hierarchy of *subgraphs* reveals nodes that are placed at the *core* of the initial graph. This is based on the assumption that keywords in an article's graph are not necessarily the ones with high number of connections or high frequency, but

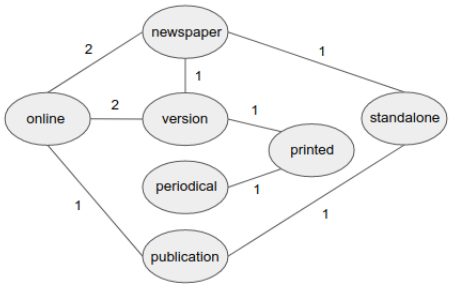

Figure 2: A graph constructed for an example article: "*Online newspaper is the online version of a newspaper, either as a standalone publication or as the online version of a printed periodical*", with a sliding window of size 2.

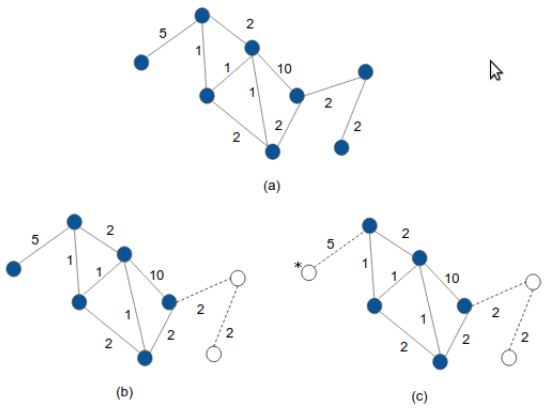

Figure 3: Graph decomposition: (a) the main graph. (b) the 3-core decomposition. (c) the 3-truss decomposition. Note that the node * is removed in the truss decomposition because its connecting edge is not part of a triangle.

the ones that are at the core of the graph and can reach many other nodes (spreaders).

There are several ways to decompose a graph into subgraphs for keyword extraction. In this work, we adopt two methods: k-core [5] and k-truss [9] graph decomposition. ***k*-core** decomposition relies

on *peeling away* weakly connected nodes to gradually get some understanding of the core of the graph. The k-core of a graph is the maximal subgraph such that every node has a degree at least k, where the degree of a node is the sum of the weights on its connecting edges. ***k*-truss** decomposition is an extension to k-core decomposition, in which edges are pruned first if they are weak, and then nodes with no more connecting edges are peeled off. k-truss basically prunes an edge from the k-1 subgraph if it is not supported by at least k-2 other edges that form triangles with that edge. Figure 3 shows an example of both decomposition methods.

In the decomposition process, both the k-core and the k-truss methods continue increasing k and prune nodes to create more *cohesive* subgraphs until they reach k-max, which is the deepest level they can reach in decomposition (after which the resulting subgraph becomes empty). A core/truss number of a node is then defined as the maximum core/truss number at which this node exists.

2.4 Keyword Extraction

Assuming that the graph was decomposed into cores/trusses, we assign a score to each node that is equal to the sum of the core/truss numbers of its neighbors in the the 0-core/0-truss graph. This scoring is based on the assumption that nodes in the same core/truss are as good as spreaders in the graph [1]. If a score is assigned instead based only on the node's own core/truss number, then many nodes will end up having the same weight. After assigning scores to nodes, keywords then can be selected using different methods. In this work, we first set the number of keywords N to be selected to P% of the number of nodes in the article's full graph. To account for short articles, we test if N is less than a predefined minimum *min*. If yes, it is set to *min*. Next, we sort the nodes given their core/truss scores and select the query keywords as the top N terms having the highest scores. A term with a weight equal to the Nth node is also selected.

2.5 Retrieving Background Articles

The set of selected keywords constructs the search query Q, where each keyword is *weighted/boosted* by its score. We issue this query against the collection index to retrieve a potential set of background articles. Finally, duplicate articles from the results (that have the same author, title and publishing date but different identifier) are removed, and the top 100 links found are reported.

3 EXPERIMENTAL EVALUATION

3.1 Dataset Preprocessing and Indexing

We used version 2 of the Washington Post news test collection released for the news track by TREC[1] in 2018. To index the collection, we used Lucene[2] ver. 8.0. During our indexing process, we excluded articles that are "Opinions", "Letters to the Editor", or "The Post's View" as they were declared by TREC to be non-relevant. The articles in this version were provided as JSON objects in a single file. For each article, we extracted the metadata (title, author, publishing date, and URL) and indexed them as separate string fields in Lucene. For the article's body, we first concatenated the HTML text contents from the JSON object (marked by a "sanitized_html" type). We then used JSOUP library[3] to extract the raw text from the HTML text. Afterwards, we lower-cased the text and removed stop words and all non-alphabetical characters. The final preprocessed text was indexed as a text field in Lucene along with the article's metadata.

For the query articles, we first extracted its title and body, concatenated both in one string, then we preprocessed it the same way like the other indexed articles. For retrieving background links, we used Lucene's default ranking model (a variation of the Vector Space Model), and we submitted the first 100 background articles retrieved (after removing duplicates).

3.2 Evaluation Measures

To evaluate the effectiveness of the proposed solutions to the background linking problem, nDCG@5 was used as a primary metric by TREC. The gain value for each retrieved background article was calculated as 2^r where r indicates how much background and context knowledge the retrieved background article provides to the query article. The gain value r ranges from 0 (the article provides little or no useful background information) to 4 (the article must appear in the sidebar otherwise critical context is missing).

As participants were asked to submit up to 100 background links for each query article, other metrics than nDCG@5 were also reported back by TREC for each submitted run (e.g., precision at different levels).

3.3 Official Runs

We experimented with the two graph decomposition methods explained in the previous section in solving the background linking problem. For each method, we tuned a number of parameters using the TREC 2018 queries and relevant judgments. The parameters we tuned were:

- The *sliding window size* used when creating the article's graph.
- The *keyword selection parameters*, which are the percentage of terms to be selected P and the minimum number *min* of keywords to be selected for short articles.
- The *title words boost factor*. After the analysis of the article and the selection of keywords to construct the query, keywords that are part of the article's title are given a boost factor that is multiplied by their weight before issuing the query to Lucene. This is driven by our belief that authors carefully select title words to mostly reflect the message/story of the article.

As stated in the guidelines for this task last year and also this year, the user is assumed to read the query article in the present time. This allows articles published at any time to be candidate background articles to the query article. Nonetheless, during our tuning and testing phase using the TREC 2018 set of query articles and relevant judgements, we always found that excluding articles that were published after the query article (denoted as forward links) from the results exhibits better performance. Therefore, for each of the graph decomposition methods we used for analyzing

[1] https://trec.nist.gov/data/wapost/
[2] http://lucene.apache.org/
[3] https://jsoup.org/

the query article, we submitted two runs for TREC 2019: one that does not include forward links, and another that does. The runs we submitted were as follows:

- *QU_KCore*: In this run, we used k-core as our graph decomposition method with a window size set to 4, the percentage of terms P set to 20% , *min* for short articles set to 70, and a title boost set to 1.7. In this run, we excluded forward links from the results.
- *QU_KCore_F*: In this run, we used the same settings as in *QU_KCore*, but we included forward links in the results.
- *QU_KTruss*: In this run, we used k-truss as our graph decomposition method with a window size set to 6, the percentage of terms P set to 20%, and *min* for short articles set to 70. We also excluded forward links from the results.
- *QU_KTruss_F*: In this run, we used the same settings as in *QU_KTruss*, but we included forward links in the results.

3.4 Experimental Results

Table 1 shows the nDCG@5 values achieved by our methods. In general, all our runs outperform the TREC median for this task with our best run, (*QU_KCore*), ranked second among 28 total submitted runs. This supports, to a great extent, the effectiveness of using graph-based text analysis in solving the news background linking problem. We also notice that using the k-core method for the article's graph analysis and decomposition produced a higher nDCG@5 score compared to k-truss.

We can also notice from Table 1, as expected, that the runs that excluded forward links produced better results using both graph decomposition methods. Nonetheless, as mentioned before, other metrics than nDCG@5 were also measured. In our analysis of the precision scores that our methods achieved at different cutoffs, we found that, as shown in Figure 4, including the forward links to the results set generally yielded retrieval of more relevant articles. This measure of performance, however, ignores how much useful is the retrieved background article to the query article.

Run	nDCG@5	Rank
TREC'19-Median	0.5295	-
TREC'19-Best	0.6064	1st
QU_KCore	**0.5918**	2nd
QU_KTruss	0.5807	3rd
QU_KCore_F	0.5723	7th
QU_KTruss_F	0.5689	9th

Table 1: nDCG@5 scores for the submitted runs compared to TREC'19 median and best runs.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we described the approach we adopted to tackle the problem of news background linking as part of our participation in TREC 2019 news track. Our solution relied basically on extracting representative keywords from the news query article, and using them to find the article's background links. We used a graph-based analysis of articles for keyword extraction, and varied the methods used for constructing the graph and analyzing it. The official results

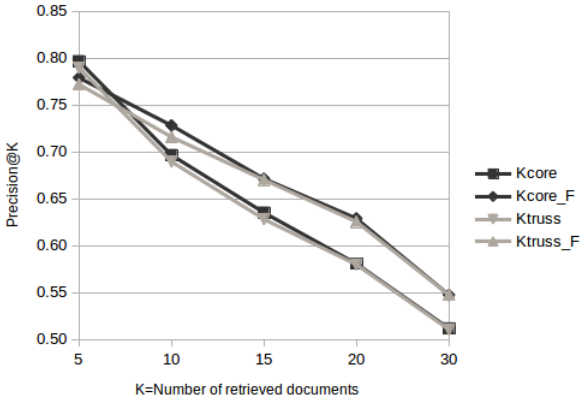

Figure 4: The precision scores obtained for all the submitted runs at different numbers of retrieved documents.

reported by TREC showed that our proposed approach, in different settings, outperformed the TREC median for this task. Moreover, our best submitted run was ranked second among 28 total submitted runs to the task, indicating the potential effectiveness of our approach. Our future work includes conducting more deep experiments to confirm the effectiveness of adopting graph-based text analysis in solving the news background linking problem. We will also investigate the idea of expanding the extracted keywords using other methods such as word embedding similarities.

ACKNOWLEDGMENTS

This work was made possible by NPRP grant# NPRP 11S-1204-170060 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

REFERENCES

[1] Joonhyun Bae and Sangwook Kim. 2014. Identifying and ranking influential spreaders in complex networks by neighborhood coreness. *Physica A: Statistical Mechanics and its Applications* 395 (2014), 549–559.
[2] Agra Bimantara, Michelle Blau, Kevin Engelhardt, Johannes Gerwert, Tobias Gottschalk, Philipp Lukosz, Shenna Piri, Nima Saken Shaft, and Klaus Berberich. 2018. htw saar @ TREC 2018 News Track. (2018).
[3] Pilar Lopez-Ubeda, Manuel Carlos Diaz-Galiano, Maria Teresa Martin Valdivia, and L. Alfonso Urena-Lopez. 2018. Using clustering to filter results of an Information Retrieval system. (2018).
[4] Kuang Lua and Hui Fang. 2018. Paragraph as Lead - Finding Background Documents for News Articles. (2018).
[5] François Rousseau and Michalis Vazirgiannis. 2015. Main core retention on graph-of-words for single-document keyword extraction. In *European Conference on Information Retrieval*. Springer, 382–393.
[6] Guokan Shang, Wensi Ding, Zekun Zhang, Antoine Jean-Pierre Tixier, Polykarpos Meladianos, Michalis Vazirgiannis, and Jean-Pierre Lorré. 2018. Unsupervised Abstractive Meeting Summarization with Multi-Sentence Compression and Budgeted Submodular Maximization. *arXiv preprint arXiv:1805.05271* (2018).
[7] Konstantinos Skianis, Fragkiskos Malliaros, and Michalis Vazirgiannis. 2018. Fusing document, collection and label graph-based representations with word embeddings for text classification. In *NAACL-HLT Workshop on Graph-Based Natural Language Processing (TextGraphs)*.
[8] Ian Soboroff, Shudong Huang, and Donna Harman. 2018. TREC 2018 News Track Overview. (2018).
[9] Antoine Tixier, Fragkiskos Malliaros, and Michalis Vazirgiannis. 2016. A graph degeneracy-based approach to keyword extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1860–1870.
[10] Peilin Yang and Jimmy Lin. 2018. Anserini at TREC 2018: CENTRE, Common Core, and News Tracks. (2018).