bigIR at TREC 2021: Adopting Transfer Learning for News Background Linking

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

ABSTRACT

In this paper, we present the participation of the bigIR team at Qatar University in the TREC 2021 news track. We participated in the background linking task. The task mainly aims to retrieve news articles that provide context and background knowledge to the reader of a specific query article. We submitted five runs for this task. In the first two, we adopted an ad-hoc retrieval approach, where the query articles were analyzed to generate search queries that were issued against the news articles collection to retrieve the required links. In the remaining runs, we adopted a transfer learning approach to rerank the articles retrieved given their usefulness to address specific subtopics related to the query articles. These subtopics were given by the track organizers as a new challenge this year. The results show that one of our runs outperformed TREC median submission, while others achieved comparable results.

1 INTRODUCTION

Motivated to help readers of online news articles to better understand its content and gain more background knowledge on its context, the news background linking task was proposed by the Text REtrieval Conference (TREC) in 2018, with follow-ups in 2019, 2020 and 2021. The task mainly requires researchers to analyze an input query news article to provide links to other news articles that the reader can read to gain more context and background knowledge on the content of the input article.

Many teams participated in this news background linking task, providing different solutions to retrieve the required background articles [1, 5, 7]. Surprisingly though, the most effective method reported to date uses the whole content of the query article as a search query in an ad-hoc setting to retrieve the background links. Aside from the fact that this method is considered inefficient, as it generates very long search queries, it is still far from being optimal; it achieved nDCG@5 of 0.46, 0.63 and 0.59 in TREC 2018, 2019, and 2020 respectively [8–10], which leaves room for improvement.

In this paper, we demonstrate the participation of our bigIR team at Qatar University in the background linking task in TREC 2021. We submitted five runs for the task. One of our runs outperformed TREC median submission, while others achieved comparable results. In two runs, we adopted an ad-hoc retrieval approach, where we analyzed the query article to extract a search query that we issued against the collection of the news articles to retrieve the background links. To analyze the query articles, we adopted two keyword extraction techniques: *k*-Truss [11], which is a graph-based keyword extraction algorithm that we experimented with before and showed promising results [3], and YAKE! [2], a statistical keyword extraction technique that was recently released and achieved effective performance in keyword extraction that outperformed many well-known baselines.

For the other three runs, we adopted a transfer learning approach to rerank the articles, retrieved using the ad-hoc based approach, given their relevance to the subtopics of the query article. Our approach mainly relied on splitting the retrieved articles into passages, rerank those passages, using pre-trained transformer models, given their relevance to the subtopics description, then use the passage scores to assign article scores to rerank the news articles. We experimented with two pre-trained BERT-based reranker models to rerank the passages: *monoBERT* [6], a relevance classifier model for query-passage pairs, and *DistilBERT-TAS-B* [4], a sentence transformer model that is optimized for the task of semantic search. Our results show that using the monoBERT model is better for the background linking task.

The rest of this paper is organized as follows. We provide details on the dataset released for this task and how we processed it in Section 2. Details on how we generated the main news background linking runs are discussed in Section 3. Details on how we applied transfer learning to generate the subtopic runs are given in Section 4. Results are presented in Section 5. Finally, our conclusion and future work are presented in Section 6.

2 DATASET AND PREPROCESSING

We used version 4 of the Washington Post news collection released for the news track by TREC[1] in 2021. The collection contains 728,626 news articles, published by the Washington Post, that span nine years from 2012 to 2020. Within the dataset file, the news articles are written as JSON objects. For each article, we extracted and stored the metadata (title, author, publishing date, and URL), along with the type of the article (e.g., opinions, sports, climate, politics, etc.). For the article's main content, we extracted from the JSON object only the content that is marked by a "sanitized_html" type, and ignored other descriptions of media or other embedded objects. We then used JSOUP library[2] to extract the raw text from the HTML text. We stored that text of the article, split into paragraphs (as marked in the JSON object) into a mySQL database. We also used Lucene[3] ver. 8.0 to create an inverted index of the collection for ad-hoc retrieval purposes. To index the articles, we concatenated all paragraphs, lower-cased the text, and removed stop words, all non-alphabetical characters, and all single character terms. The final preprocessed text was indexed as a text field in Lucene along with the article's metadata.

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

3 AD-HOC BASED RETRIEVAL OF BACKGROUND LINKS

We used mainly an ad-hoc based retrieval approach to generate our first two submitted runs: **QU_LeadPar** and **QU_YakeTruss**. Our approach mainly relies on extracting keywords from the article that best indicate its relevant subtopics, weigh these keywords according to its influence in the article, and then generate a weighted search query out of these keywords. The query is then issued against an index of the news article collection to retrieve the required background links. More precisely, let N be the set of extracted terms from an input query article, and assume that each term $t_i \in N$ has a weight w_{t_i}, that indicates its importance in the query article. We choose $k \subset N$ terms with the highest weights to construct the search query Q as follows:

$$Q = \{(t_1, w_{t_1}), (t_2, w_{t_2}), ..., (t_k, w_{t_k})\} \quad (1)$$

To generate our first run **QU_LeadPar**, we simply extracted the unique terms that appeared in the title of the query article plus its first 16 leading paragraphs, and used these as search terms, with their weight assigned as the term frequencies. We opted to extract the terms for this run from only the 16 leading paragraphs, as our experiments on the queries from last years showed that using 16 paragraphs on average is sufficient to achieve high effectiveness.

To generate the second run **QU_YakeTruss**, we adopted two keyword extraction methods to extract the keywords from the query article; namely k-Truss [11], a graph-based algorithm, and YAKE![2], a recent statistical algorithm. We then used linear interpolation to aggregate their assigned weights to the extracted keywords to generate the final search query as follows:

$$W_{t_i} = \alpha * WTruss_{t_i} + (1 - \alpha) * WYAKE_{t_i} \quad (2)$$

where $WTruss_{t_i}$ and $WYAKE_{t_i}$ are the weights assigned by the k-Truss and YAKE! methods respectively to the term t_i. In our run, we set the interpolation parameter $alpha$ to 0.5. After interpolating the term weights from k-Truss and YAKE!, we selected the top 100 terms to generate our search query. Both methods are briefly described below along with how we used them to assign the weights to the different terms.

3.1 *k*-Truss

k-Truss [11] is a graph decomposition technique that mainly relies on converting the text into a graph of words and analyzing this graph to reveal the most influential words/nodes. To build this graph, the unique terms in the text are added as nodes, then a sliding window goes over the text from left to right, creating edges between co-occurring terms within the window. The edges are weighted with the co-occurring frequency of the terms. k-Truss analyzes the created graph by decomposing it into a set of nested subgraphs through iteratively pruning weak edges. Precisely, k-Truss 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. A node with no more connecting edges is pruned from the graph as well. Finally, each node is assigned a truss number equal to the maximum subgraph number in which it resides. A truss number can then be used as a node's weight. However, this will result in many nodes having the same weight. Therefore, in this work, we adopted the work in [11], and assigned a weight to the node (the term) equal to the sum of truss numbers of its neighbors in the original graph of text. Figure 1 shows an example of applying k-Truss decomposition. In our run, we set the width of the sliding window to 3.

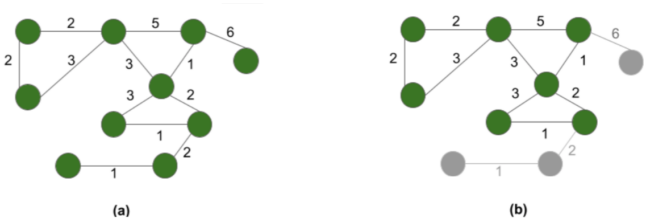

Figure 1: Decomposing a graph into subgraphs. (a) original graph (b) 3-Truss decomposition.

3.2 YAKE!

YAKE! is a statistical keyword extraction method that was proposed recently, and outperformed many well-known keyword extraction methods [2]. It mainly aggregates the following statistical features of a term in the text to calculate its weight:

- The term frequency.
- How frequent the term is mentioned starting with a capital case letter excluding the beginning of a sentence, or marked as an acronym.
- The position of the term in the document, favoring more terms that occur at the beginning of the document.
- The percentage of different sentences in which the term appears, on an assumption that terms that occur in different sentences are more influential.
- The term context, by capturing the number of different terms that co-occur with the term on both its left and right sides, on the assumption that the higher this number is, the less significant the term will be.

After aggregating YAKE! features, and as reported by its authors according to the aggregation equation[2], the smaller the YAKE! value of a term is, the more significant it is. Since in our work we assign weights to terms that reflect their influence, we convert this score to a boost score by taking its reciprocal value.

3.3 Background Links Retrieval

For retrieving the background links, we used Lucene's default ranking model. We used the "type" field from the retrieved articles to filter out the ones that are "Opinions", "Letters to the Editor", or "The Post's View", as they were declared by TREC to be non-relevant. We also filtered out all articles that were published after the query article. Finally, we submitted the first 100 background articles retrieved (after removing duplicates).

4 TRANSFER LEARNING FOR NEWS BACKGROUND LINKING

The news background linking task this year involved the new submission of "subtopic" runs. In these runs, the researchers are asked

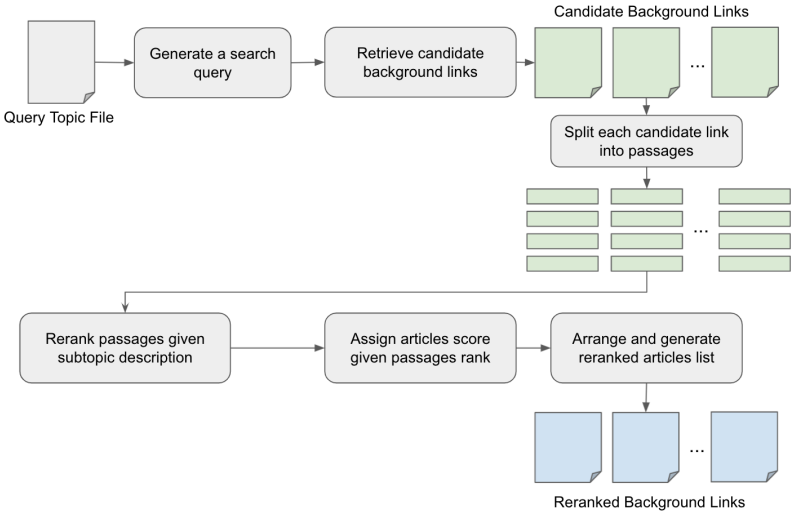

Figure 2: A high level overview of the background links retrieval system used for generating the "subtopic" runs

to retrieve news articles that address a specific subtopic that is either: 1) mentioned in the query article; thus the background article is supposed to give more details on the subtopic, or 2) not mentioned in the query article, but somehow related to it; thus reading about it will allow the query article's reader to gain more knowledge on its context. The description of the subtopics for each query article was given by the organizers of the track within the description of the query article topics, and participants were only allowed to use this description to generate "subtopic" runs. In our participation for the background linking task, we generated three "subtopic" runs. In this section, we first describe our general approach to generate these runs, then we discuss each of them.

4.1 Approach Overview

To generate the "subtopic" runs, we first used a search query extracted from the query article to retrieve an initial set of 100 candidate background links from the news articles collection. As with our first two runs, we filtered out irrelevant articles and articles that were published after the query article. We then adopted a transfer learning approach to rerank this candidate set, given the description of each subtopic. More precisely, we adopted two pretrained BERT-based reranker models to rerank the candidate background links given the subtopics description. The models we used are: *monoBERT* [6], which is a relevance classifier model that was trained on the MS MARCO dataset,[4] ranking document passages given how they answer an input question, and *DistilBERT-TAS-B* [4], whic is also trained on MS MARCO for passage reranking, but with balanced topic-aware sampling.

Since both models were trained on reranking passages, which is of small size compared to the long news articles, we split each candidate background article into passages, and feed those passages along with the subtopic description as a query to the reranker model. This is to avoid important text truncation if the articles are fed directly along with the subtopics to the reranker model. For simplicity, we consider each paragraph in the candidate article as a single passage, and any paragraph that is less than 20 terms in length is concatenated with its predecessor in a single passage. After reranking the passages, we assign each candidate article a score equal to the maximum relevance score assigned to any of its passages. Finally, we rerank the candidate articles given their scores. Figure 2 shows an outline of the adopted retrieval approach.

[4]https://microsoft.github.io/msmarco/

4.2 Submitted Subtopic Runs

Our submitted runs are described as follows:

- **QU_SP_MBERT**: In this run, we obtained initially the set of candidate background links using the 16 leading paragraph of the query article along with its title as a search query, and monoBERT was used as a reranker model.
- **QU_SP_MSM**: In this run, we obtained initially the set of candidate background links using the 16 leading paragraph of the query article along with its title as a search query, and DistilBERT-TAS-B was used as a reranker model.
- **QU_SS_MSM**: In this run, we obtained initially the set of candidate background links using a concatenation of all subtopics of the query article along with its title as a search query, and DistilBERT-TAS-B was used as a reranker model.

5 OFFICIAL TREC RESULTS

Table 1 shows the results as provided to us by the track organizers for the main background linking task. In this task, nDCG@10 was used as the official evaluation metric this year. As can be seen, **QU_leadPar** outperformed the TREC median for this task. This result, as in previous years of the track, shows again the effectiveness of using almost the full article as a search query to retrieve the background links. **QU_YakeTruss** did not achieve similar results, eliminating the need for the extensive keyword extraction process, at least for effectiveness purposes.

Table 2 shows the results for the subtopic runs. For these runs, the organizers reported also the Precision@10 evaluation metric.

Run	nDCG@10
TREC'21-Median	0.3581
QU_leadPar	**0.3774**
QU_YakeTruss	0.3418

Table 1: nDCG@10 scores for the submitted runs for the main background linking task compared to TREC'21 median

Run	nDCG@10	Precision@10
TREC'21-Median	0.2177	0.1974
QU_SP_MBERT	0.2176	0.1956
QU_SP_MSM	0.1983	0.1674
QU_SS_MSM	0.2019	0.1815

Table 2: Effectiveness scores for the submitted runs for the subtopic runs of the background linking task compared to TREC'21 median

We clearly notice that the results are generally much lower than the main background linking task, indicating the challenging nature for this specific type of news background linking. It can be also noticed that using **monoBERT** for reranking the passages in this task is generally better than using **DistilBERT-TAS-B**, as its performance is comparable to the TREC median in both evaluation metrics.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we present our bigIR group's participation in the background linking task in TREC 2021. This year, we adopted two approached to tackle the problem of news background linking. The first is ad-hoc retrieval based, which depends on the extraction of a search query from the input article to retrieve the required links. The results showed that using the most frequent items extracted from the 16 leading paragraphs of the query article achieved the best results among our submitted runs. The second approach generated subtopics runs, which mainly depends on using BERT-based reranker models to rerank the retrieved articles given their relevance to the described subtopic. Our results for those runs showed that using monoBERT as a reranker model achieved comparable results to the TREC median for this task. The results also showed that the effectiveness for those runs is generally much lower than the main background linking task, which leaves a big room for improvement in the future.

Our future work will initially focus on investigating the effect of the different parameters that we adopted in the design of our approach on the results achieved by our runs. For instance, we used the date filter to filter out articles that were published after the query article, and we believe that we might have missed relevant articles this way, specially if the query article was published early in the news collection. We will also work on addressing the subtopic-based news linking problem as it is challenging. We aim to investigate other models that best represent the candidate news articles along with the subtopics for reranking.

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] 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. In *Proceedings of the Twenty-Seventh Text REtrieval Conference (TREC)*.

[2] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alipio Jorge, Célia Nunes, and Adam Jatowt. 2020. YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences* 509 (2020), 257–289.

[3] Marwa Essam and Tamer Elsayed. 2019. bigIR at TREC 2019: Graph-based Analysis for News Background Linking.. In *Proceedings of the Twenty-Eighth Text REtrieval Conference (TREC)*.

[4] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *Proc. of SIGIR*.

[5] Kuang Lu and Hui Fang. 2019. Leveraging Entities in Background Document Retrieval for News Articles. In *Proceedings of the Twenty-Eighth Text REtrieval Conference (TREC)*.

[6] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[7] Dwaipayan Roy Rahul Gautam, Mandar Mitra. 2020. TREC 2020 NEWS Track Background Linking Task. In *Proceedings of the Twenty-Ninth Text REtrieval Conference (TREC)*.

[8] Ian Soboroff, Shudong Huang, and Donna Harman. 2018. TREC 2018 News Track Overview. In *Proceedings of the Twenty-Seventh Text REtrieval Conference (TREC)*.

[9] Ian Soboroff, Shudong Huang, and Donna Harman. 2019. TREC 2019 News Track Overview. In *Proceedings of the Twenty-Eighth Text REtrieval Conference (TREC)*.

[10] Ian Soboroff, Shudong Huang, and Donna Harman. 2020. TREC 2020 News Track Overview. In *Proceedings of the Twenty-Ninth Text REtrieval Conference (TREC)*.

[11] 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.