Paragraph as Lead - Finding Background Documents for News Articles

Kuang Lu[a] and Hui Fang[a]

[a]University of Delaware, Newark DE 19716, USA

Abstract

When reading a news article, it is very useful that articles about the background of various aspects of the story are provided so that the readers can better understand the story. In this year's News Track, we tried to use paragraphs as leads to find background articles since they tend to cover different aspects of the main story [3]. More specifically, the keywords in the paragraphs were extracted and used as queries to find background articles. Entities were also leveraged to improve the retrieval performances of the keyword queries.

1 Introduction

The News track is introduced this year with the goal of identifying the search needs of readers and journalists [1]. Two potential needs are identified. The first need is, for a news article, to link a list of other articles that a user should read next in order to get the context and background information of the article. This would help readers better understand the story and evaluate the trustworthiness it. The second need is identified as ranking the entities in a news piece based on whether the external information, such as Wikipedia pages, would help readers better understand the piece.

Based on these needs, two tasks are introduced for the news track, which are background linking and entity ranking. In this year's News Track, we focus on the first task and propose a method that uses paragraphs as leads to find the background articles. More specifically, for each query article, we extract paragraphs for them and identify keywords from the paragraphs using a probabilistic mode. The keywords are used to retrieve articles related to the paragraphs. The retrieval process is improved by using entities to reduce ambiguity. The articles retrieved for different paragraphs are then merged and ranked to generate the final background article list.

[1]trec-news.org

2 Paragraph-Based Background Article Discovery

As mentioned before, we used paragraphs as leads to find background articles. This method is proposed based on our understanding of what background information a reader needs and the unique structure of news articles. A news article usually has a main story as well as its different aspects. An example would be for a news piece about an earthquake, the event itself is reported and other aspects of it such as its aftermath, disaster relief efforts, and earthquake history of the region are also likely to be covered. The background articles for this article could be some other general reports of the same event, or articles that cover in details the aspects mentioned in the original article. Thus, these aspects as well as the main story can be used as leads to find the background articles. Then the problem becomes to how they can be extracted from the article. Luckily, the structure of the news may certainly accelerate such process. According to Fox [3], a news article often starts with a summary of the main story, following by detailed coverage of major aspects of the story. The aspects are reported in the order of importance and the unity and completeness of the writing of the aspects are ensured so that even latter an editor decides to cutoff a part of the article from the end, which are aspects that are least important, the story itself will not damaged. Therefore, logical breaks can be inserted into the article so that it is divided into the coverage of various aspects of the article. These aspects then can be used as leads to find background articles. The physical breaks between paragraphs, we argue, can serve as the logic breaks. The information inside a paragraph usually is about only one aspect of the story and therefore the breaks between aspects can only happen at the breaks between paragraphs. As a result, paragraphs can be treated as a text unit of an aspect of the main story that we can use to find background articles about.

It is important to note that the report of an aspect of the main story can span multiple paragraphs [3]. However, we argue that paragraphs about the same aspect will result in similar background articles, and through the final merging and ranking step, this redundancy may not be a problem.

Having paragraphs as leads for finding background articles, the next question that needs to be answered is how should the leads be used. We argue that the idea of relatedness between a paragraph and a background article is similar as the relevance concept in IR. Therefore, it is tackled as a retrieval problem. However, paragraphs are often too long and therefore directly using them for retrieval might lead to noisy results, as it is observed for verbose query retrieval [4]. Thus, we need a way to find keywords, which are the words that are most representative of the meaning of the paragraphs, so that they can instead be used as the queries. We define keywords as the words that maximize the probability of $P(p|w)$ in which p is a paragraph and w is a word in the paragraph. This is essentially the generation probability of a paragraph given a word. Using Bayes' theorem and bag-of-words assumption it can be re-written as:

$$P(p|w) = \frac{P(w|p)P(p)}{P(w)} \propto P(w|p) = \sum_{w_i \in p} P(w|w_i) \quad (1)$$

where we assume $P(w_i)$ follows a uniform distribution and $\propto$ denotes rank equivalent. $P(w|w_i)$ can be easily computed using word co-occurrence and therefore this equation can help us effectively extract keywords.

3 Submitted Runs

In this section, we describe how the data was processed and how our runs for the background linking task were generated.

3.1 Data Processing

In this year's News Track, the collection is shared by Washington Post, which contains five years of their news articles from 2012 to 2017. All articles in the collection were processed in the same way in which only document id, title, timestamp, and the text of each paragraph were kept. The text of paragraphs of the same article is merged to form a "body" field whereas the title of the article is used as the "title" field of the document. These two fields were then used for indexing and retrieval. A document was constructed in this way so that the "title" field can be prioritized in the retrieval phase. Using the documents, two different types of indices were created in terms of how the text was tokenized. The first type of index tokenized the text normally and used space as the separators of different terms. However, we treated entities, either single-term or multi-term, as single words for the second index type. Entities have been used for improving the retrieval performances, since they are less ambiguous and generally more important than other terms in queries [5]. More specifically, the entities in the text were identified using DBpedia Spotlight [1] which automatically annotates DBpedia entities from text. The entities were subsequently replaced by their canonical forms (e.g. the form appears in DBpedia), which are also provided by the toolkit. For example, "the Red Planet" and "Mars" are all mapped to the canonical form "Mars". These canonical forms were used as a single word for indexing. We hope that such way of indexing would help us to more accurately match the entities in the query articles and other articles, which may certainly leads to more effective background article discovery. The parameter "Confidence" of the tool is set to 0.5.

For query articles, document id, timestamp and the text of each paragraph were kept. However, the paragraph text is not merged since the paragraphs were used separately for retrieving background articles. The text was tokenized using the same two ways as the text of other articles was tokenized. This resulted in two query types. Different query types were used with the corresponding index types. These two types are referred to as *Normal* and *Entity* for the rest of the paper.

3.2 Background Article Retrieval and Result Merging

Keywords were extracted from the two types of queries using Equation 1. F2exp [2] was employed to retrieval background articles with keywords from the paragraphs of queries. More specifically, the "title" field of article was given the weight of 0.7 and the "body" field was given the weight of 0.3. The score of a document is the sum of the scores for the two fields. We only retrieved 5 documents for each paragraph. This number as well as the field weights were set empirically according to the test results on the three example queries provided by the organizers.

After retrieving documents, the final step is to merge the results of each paragraph. We used three ways of merging results. The first one is simply to use the maximum score a document receives among different paragraphs to rank the documents. We call this method *score-order*. The second one is to use the "voting" of the paragraphs. A document that appears in more paragraphs' results is ranked higher. If there is a vote tie, it is broken by the highest scores that the document receives. This way of ranking documents is called *vote-order*. The third way of merging results ranks the top documents (e.g. the ones that ranked first) for the results of different paragraphs based on their scores. This process is repeated for other ranking positions from 2 to 5. The final results is the concatenation of the results of this process and the sub-result for the higher ranking positions are put higher in the final list. This merging method is named *ladder-order*.

3.3 Run Description

As discussed earlier, there are two variants of our system, which are the index/query type and the result merging method. We generated 5 runs based on these variants to investigate the effects of them.

- **UDInfolab_kweh:** Entity index and queries were used to retrieval documents and score-order was used for merging.
- **UDInfolab_kwh:** Normal index and queries were used to retrieval documents and score-order was used for merging.
- **UDInfolab_kwef:** Entity index and queries were used to retrieval documents and ladder-order was used for merging.
- **UDInfolab_kwf:** Normal index and queries were used to retrieval documents and ladder-order was used for merging.
- **UDInfolab_kwev:** Entity index and queries were used to retrieval documents and vote-order was used for merging.

Table 1: NDCG@5 for submitted runs and TREC median

Run	**NDCG@5**
$UDInfolab\_kweh$	0.2511
$UDInfolab\_kwh$	0.1818
$DInfolab\_kwef$	0.1938
$DInfolab\_kwf$	0.1444
$DInfolab\_kwev$	**0.2693**
$TREC\ median$	0.2792

4 Results and Analysis

The effectiveness of the submitted runs as well as the TREC median are reported in Table 1 as NDCG@5. When only comparing submitted runs there are two observations. First, our hypothesis that using entity helps identifying background articles seems to be supported. When the merging methods are the same, runs using *Entity* index and queries (UDInfolab_kweh, UDInfolab_kwef) outperform their counterparts using *Normal* index and queries (UDInfolab_kwh, UDInfolab_kwf). On the other hand, when comparing result merging methods, it seems that the vote-order method is the best. However, when comparing to TREC median, all of the submitted runs do not seem to perform well.

After obtaining the judgement data, we conducted experiments and error analysis to further investigate our methods. First, we investigate the core part of our runs, which is to use paragraph as lead to find background articles. More specifically, we gathered the retrieval results from the first retrieval step and computed the best possible NDCG@5 can be obtained from this pool of articles. The optimal NDCG@5 for using *Entity* and *Normal* are 0.7156 and 0.6552 respectively. Moreover, 10 out 50 queries received perfect NDCG@5 when *Entity* index and queries were used. Another important detail of the article pool is that since only 5 articles were retrieved for each paragraph, in average the pool size for a query document is only 119. Therefore, it can be concluded that our paragraph-based method can identify background articles effectively.

We then performed error analysis on the document pools mentioned above to discover why in some cases our methods failed. Two causes were discovered. First, same weights were given for each keyword of a paragraph, which leads to topic drifting. Second, we found that when computing keywords, only paragraphs were taken into account. However, a background article should be related to not only the paragraph, but also the whole document. The error analysis shows potential ways of improving our methods.

5 Conclusion

In this year's News Track, we investigated using keywords in different paragraphs to find background news articles. Moreover, the effectiveness of named entities and different result merging techniques were also tested. The effectiveness of our runs are not very satisfying. However, for article retrieval step, analysis illustrates that using paragraphs can effectively identify background articles. Limitations of the keyword generation and keyword weighting were also discovered through error analysis, which can guide us to improve the proposed methods. Besides, result merging was done using very simple method. It is reasonable to anticipate better results if more sophisticated methods are used, such considering paragraph importance of the query document.

References

[1] Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems (I-Semantics) (2013)

[2] Fang, H., Zhai, C.: An exploration of axiomatic approaches to information retrieval. In: Proceedings of SIGIR '05. SIGIR '05 (2005)

[3] Fox, W.: Writing the News. Iowa State University Press, 3 edn. (2001)

[4] Gupta, M., Bendersky, M.: Information retrieval with verbose queries. ACM - Association for Computing Machinery (March 2015)

[5] Liu, X., Fang, H.: Latent Entity Space: A Novel Retrieval Approach for Entity-Bearing Queries. Information Retrieval 18(6), 473–503 (2015)