UNC SILS at TREC 2019 News Track

Jiaming Qu and Yue Wang

School of Information and Library Science
University of North Carolina at Chapel Hill
Chapel Hill, NC
`jiaming@ad.unc.edu, wangyue@email.unc.edu`

Abstract. This paper describes our participation in the Background Linking task of TREC 2019 News Track. Our approach has two directions. After processing the corpus, we use Lucene to index and run the initial retrieval. Then we leverage the learning-to-rank idea to train a re-ranker using the 2018 relevance judgement files as ground truth, and the re-ranker is applied to the initial retrieval results to generate a new ranked list. Experiment results prove that the re-ranker significantly improves the retrieval performance (NDCG@5) compared to the initial retrieval results without the re-ranking step.

1 Introduction

The News Track started in TREC 2018, and it is the second year for this track. This track consists of two tasks: Background Linking (BL) and Entity Ranking (ER). The BL task aims to find relevant news from the whole corpus given a target news, which provides background information or further understanding for readers. The ER task aims to rank the named entity in the news according to their importance in the context, which highlights the most significant entities.

The School of Information and Library Science (SILS) at the University of North Carolina at Chapel Hill (UNC) participated in the BL task in this year's News Track. In this paper, we discuss our strategy to tackle the problem. In section 2, we provide an overview of the dataset and how we process the text. In section 3, we demonstrate in detail our retrieval strategy, especially the feature generation to train the re-ranker. In section 4, we report the retrieval performances evaluated by TREC. In section 5, we summarize our work and propose potential improvements and directions for future BL tasks or relevant research.

2 Data Processing

This year's News Track continues to use the Washington Post corpus as in 2018, which consists of more than 590,000 news articles published by The Washington Post. Even though TREC provides a version without duplication, we still notice that there remain some duplicate articles in the corpus. However, we do not remove duplicate articles as the first step of pre-processing, as recognizing duplicates for all articles in the corpus will involve significant computation. Since the main task is news retrieval, we leave news deduplication to the final final ranking step (Section 3.3).

First of all, we parse the original "JSON-line" format file into separate JSON files, with each file representing a piece of news in the corpus. Secondly, for each JSON file, we parse the file by extracting four fields which are ID, title, contents, and category, and discard other information. Note that the publishing date is no longer useful, as is officially stated in last year's overview [1] that we are assuming the user is reading the news now, irrespective of when the news was published.

When parsing the news content which is a list of paragraphs, in last year's News Track, Lu and Fang [2] propose treating each paragraph separately to generate keywords for retrieval. However, we simply concatenate each paragraph and treat the merged text as news content. When processing news titles and contents, we remove HTML tags, punctuation and stop words. During processing, we also remove news with types of "Opinion", "Letters to the Editor", or "The Post's View", for they are identified as irrelevant to the task [1], which finally leaves 571,963 news in the corpus.

Since the BL task aims to find relevant news given a target one, we assume that relevance judgement should be not only on the text level, but also on the context level. We do not take the Deep Learning approach to learn the contexts in this task, instead we simply assume that two news are more likely to be contextually relevant if they share more named entities in common. Therefore, before cleaning the text as mentioned above, we use *Spacy* [3], which is a free open-source library to do the Named Entity Recognition (NER) task to recognize entities from the news titles and contents. We discard eight kinds of entities which are "LANGUAGE", "DATE", "TIME", "PERCENT", "MONEY", "QUANTITY", "ORDINAL" and "CARDINAL", as we identify these entities, e.g., "May 1st", "250 million US dollars", "154" are not helpful to represent the context. We kept the rest kinds of entities, such as "ORG", "PERSON", "EVENT", and etc[1].

To sum up, we parse the Washington Post corpus by dividing it into separate JSON files, extracting four useful fields, then recognizing named entities from titles and contents with *Spacy*, finally cleaning the text and removing irrelevant ones.

3 Methodology

In this section, we provide detailed demonstration of our approach and intuitions behind, as is shown in Figure 1. In general, we use Apache Lucene[2], which is an open-source, high-performance and publicly-available searching engine toolkit in Java to index the corpus and run the initial searching which retrieves 1,000 documents per query. Then we generate features for each *query, document* pairs, and leverage the 2018 BL task relevance judgements to train a re-ranker. Finally

[1] https://spacy.io/api/annotation
[2] http://lucene.apache.org/

for the 2019 topics, we initially retrieve 1,000 documents for each query and re-rank the result using the trained re-ranker.

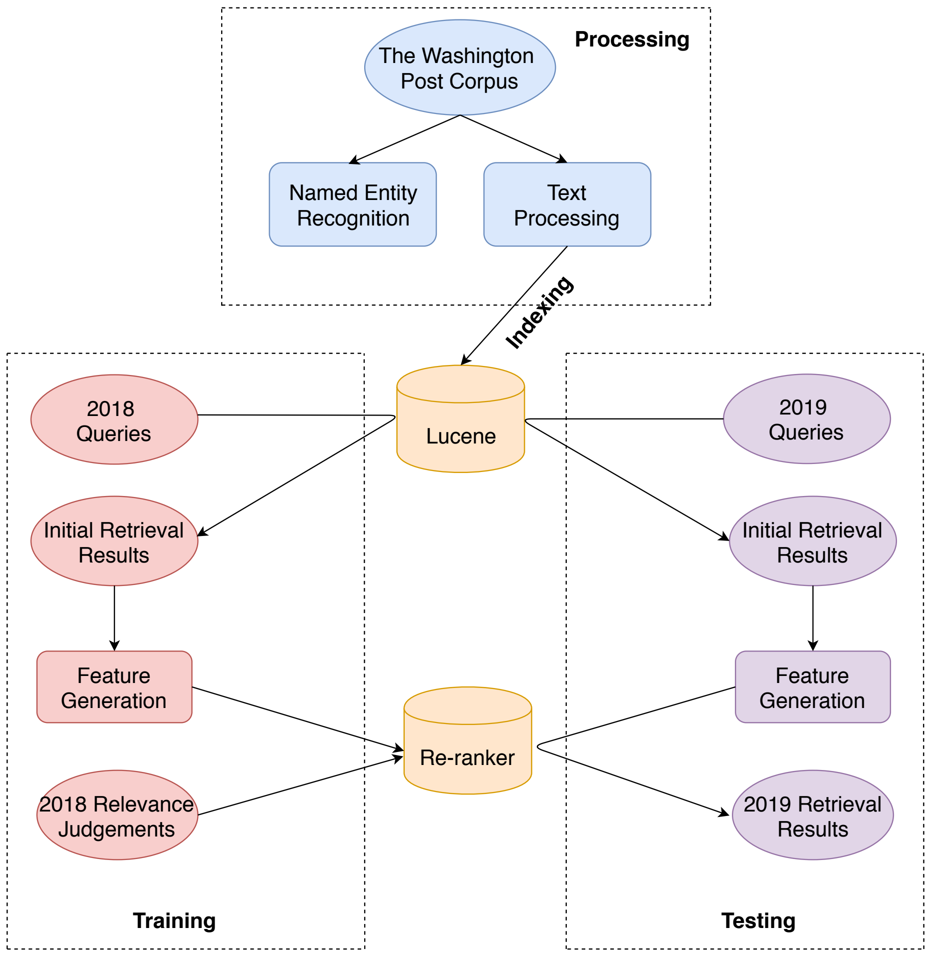


Fig. 1. General Framework

3.1 Indexing

For each news article, four fields are extracted as mentioned above, which are title, content, ID and category. For titles and contents, we use the standard analyzer built in Lucene to analyze and stem the text. We do not process ID but simply treat it as identifiers, and the category field is not used at this step but at re-ranking instead.

3.2 Initial Retrieval

One problem during the initial retrieval is how to generate the query. Instead of throwing in the whole content as an extremely long query, we extract top-K keywords with the highest TF-IDF weights as query terms. We then tune the parameter K using the 2018 data and find K=80 gives the highest NDCG@5 score. When run the search in Lucene, we use the OR operator between all the terms, and use the SHOULD operator for both the title and content field.

3.3 Re-ranking

To re-rank the initial retrieval results, we train a re-ranker based on the 2018 relevance judgements. This is a ranking problem because each *query, document* pair will have a relevance score to be sorted. We transform this ranking problem to a multi-class classification problem [4], and calculate the score of each query-document pair (q, d) as:

$$\text{score}(q, d) = \sum_{r \in \{0,2,4,8,16\}} w_r \cdot p_\theta(y = r | q, d) \tag{1}$$

where y is predicted relevance grade using a multiclass classifier $p_\theta(\cdot|q, d)$. The idea is that we treat relevance grades $r = 16, 8, 4, 2, 0$ as class labels without order, instead of its original representing relevance order. w_r are weights for each relevance grade r in the scoring function; higher weights should be assigned to higher relevance grade. Here we set $w_0 = 0, w_2 = 1, w_4 = 2, w_8 = 3, w_{16} = 4$. The formula shows that we sum up the product of a weight of a relevance grade and the probability of that relevance grade. Afterwards, we train an one-vs-all Logistic Regression classifier to do the multi-class classification.

We generate five features to train the classifier, which are *similarity of title*, *similarity of content*, *similarity of the first 100 words*, *similarity of mentions*, *similarity of category*. Below are detailed demonstration of these features.

similarity of title We construct a corpus of all the news titles, then vectorize each instance into TF-IDF vectors and measure *similarity of title* by the Cosine similarity between two vectors.

similarity of content We construct a corpus of all the news contents, then vectorize ach instance into TF-IDF vectors and measure *similarity of content* by the Cosine similarity between two vectors.

similarity of the first 100 words We construct a corpus of all the news' first 100 words, then vectorize ach instance into TF-IDF vectors and measure *similarity of the first 100 words* by the Cosine similarity between two vectors. The reason of using the first 100 words is that we assume the introduction part of a news has covered the main idea or main entities in the content.

similarity of mentions As is mentioned in Section 2, for each news we use Spacy to extract meaningful entities. Then for each pair of news, we measure *similarity of mentions* by the Jaccard similarity between two entity sets.

similarity of category In the original JSON-line file, some news have a sub-field in its content which indicates the category it belongs to, e.g., “Europe”, “Technology”, “Sports”, “Movies” and etc. There are 179 categories in total, but some news do not have such a sub-filed. To obtain a probability distribution of categories as well as to fill missing categories for each news, we generate text features from news contents, and use the basic bag-of-words model with unigrams to train a multi-class classifier, by which we predict the probability distribution of categories for each news. We measure the *similarity of category* between two news articles d_1, d_2 by the total variation of the two predicted news category distributions:

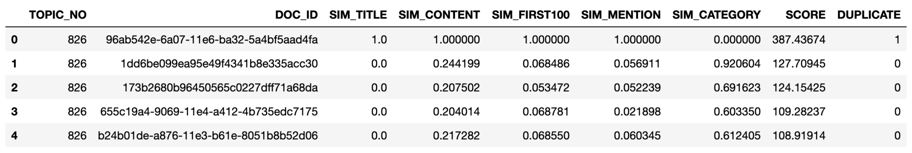
$$\delta(d_1, d_2) = \frac{1}{2} \cdot \sum_{c \in \mathcal{C}} |p_\phi(c|d_1) - p_\phi(c|d_2)| \tag{2}$$

where $\mathcal{C}$ is the set of all news categories; $p_\phi(c|d)$ is the predicted probability of article d having category c; ϕ is the parameter of the multiclass classifier. The formula shows that we sum up the absolute value of element-wise differences, and divide the sum by 2. One nice property of this metric is that it is bounded in $[0, 1]$, which is on the same scale as the cosine similarity and Jaccard similarity. Note that since it actually measures the distance, thus the higher the total variation is, the lower the similarity, which is in contrast of the cosine and Jaccard similarity.

We generate these five features for each *query, document* pair from the initial retrieval, and use the ground truth from 2018 as labels to train a multi-class classifier. We apply the same approach on the 2019 retrieval results, which could be viewed as a test set. Based on the five features generated, the re-ranker predicts a probability distribution which is converted to a re-ranking score. We test three methods using the new (re-ranking) score to re-rank. The first is purely based on the new score. The second is to normalize the raw score from Lucene and the re-ranking score separately using the min-max scaler to get a number between 0 and 1, and then add up two scores as a final score. The third is to assign different weights to the raw and re-ranking scores when adding up, and based on the ground truth from 2018, we find the optimal weight is 9 for the raw and 1 for the new. The evaluations of these three methods are shown in Section 4.

Another important contribution of *similarity of title*, *similarity of content*, *similarity of the first 100 words* is to find duplicate news. As is discussed in Section 2, there are some duplicate news in the corpus, but it is impossible to check every pair to find duplicate ones. Thus, at the re-ranking step, we identify a retrieved news is a duplicate to the target news if any of *similarity of title*, *similarity of content*, *similarity of the first 100 words*, or *similarity of mentions* is 1, as the first row shown in Figure 2. Then we simply give a final score of 0 to a duplicate news.

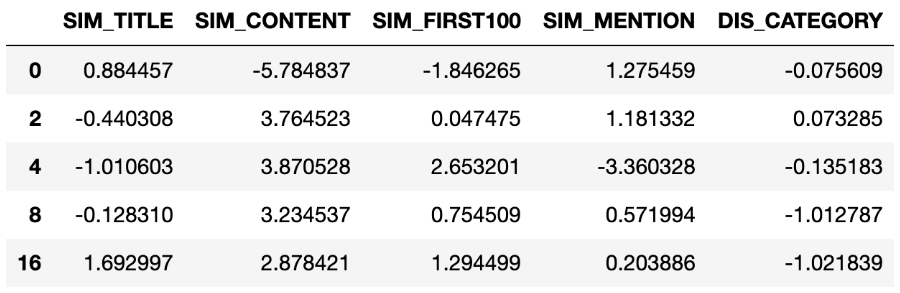
The coefficients in the multi-class Logistic Regression are shown in Figure 3. In general, we could see that ***similarity of content*** is the most important feature for each label, but different labels have a different order of feature importance. Take two extremes as an example, we can see that content similarity and first 100 words similarity features positively correlate with being perfectly relevant (label



	TOPIC_NO	DOC_ID	SIM_TITLE	SIM_CONTENT	SIM_FIRST100	SIM_MENTION	SIM_CATEGORY	SCORE	DUPLICATE
0	826	96ab542e-6a07-11e6-ba32-5a4bf5aad4fa	1.0	1.000000	1.000000	1.000000	0.000000	387.43674	1
1	826	1dd6be099ea95e49f4341b8e335acc30	0.0	0.244199	0.068486	0.056911	0.920604	127.70945	0
2	826	173b2680b96450565c0227dff71a68da	0.0	0.207502	0.053472	0.052239	0.691623	124.15425	0
3	826	655c19a4-9069-11e4-a412-4b735edc7175	0.0	0.204014	0.068781	0.021898	0.603350	109.28237	0
4	826	b24b01de-a876-11e3-b61e-8051b8b52d06	0.0	0.217282	0.068550	0.060345	0.612405	108.91914	0

Fig. 2. Feature Table

= 16), while negatively correlate with being non-relevant (label = 0), which makes sense. However we found that not all learned coefficients are easily interpretable.



	SIM_TITLE	SIM_CONTENT	SIM_FIRST100	SIM_MENTION	DIS_CATEGORY
0	0.884457	-5.784837	-1.846265	1.275459	-0.075609
2	-0.440308	3.764523	0.047475	1.181332	0.073285
4	-1.010603	3.870528	2.653201	-3.360328	-0.135183
8	-0.128310	3.234537	0.754509	0.571994	-1.012787
16	1.692997	2.878421	1.294499	0.203886	-1.021839

Fig. 3. Learned coefficients for different similarity features in the 5 one-vs-all logistic regression classifiers.

4 Evaluation

We submit 4 runs for the BL task and the evaluation results are summarized in the table below.

Table 1. Evaluation Results

Run Name	NDCG@5
sils_news_run1	0.3482
sils_news_run2	0.5502
sils_news_run3	0.5472
sils_news_run4	0.3805

sils_news_run1 is the baseline which only uses the raw BM25 score from Lucene to rank the retrieval result.

sils_news_run2 only uses the re-ranking score to rank the initial retrieval result.

sils_news_run3 uses both the raw BM25 score from Lucene and the re-ranking score by adding up to rank the initial retrieval result.

sils_news_run4 uses both the raw BM25 score from Lucene and the re-ranking score by adding up with weights to rank the initial retrieval result.

By comparing ***sils_news_run1*** and ***sils_news_run2*** which rank the retrieval results based on two different scores, we could see that the re-ranking score reflect the true ranking better than the raw BM25 score.

Also, by comparing ***sils_news_run2*** and ***sils_news_run3***, we could see that adding the raw BM25 score to the re-ranking score does not differ much from only using the re-ranking score.

Finally, by comparing ***sils_news_run3*** and ***sils_news_run4***, we could see that adding up two scores with different weights hurt the performance much, which is to our surprise because these weights improves NDCG@5 on the 2018 validation set.

5 Conclusion

We participate in the 2019 News track and submit 4 runs for the Background Linking task. Our best run incorporates two steps, which are the initial retrieval and re-ranking. We use Lucene to run the initial retrieval, and then leverage the learning-to-rank idea to train a classifier to re-rank the initial retrieval result. We convert the ranking problem to a classification problem and generate five features from both text and context. Results show that the learnt re-ranker improves performance in terms of NDCG@5 compared to the baseline model without re-ranking.

In future work, we aim to find a better approach to generate the query which maximize the performance at the initial retrieval, as the re-ranking will not be so effective if there are few relevant news in the initial retrieval result. Also, we aim to do feature engineering to generate better both high-level and query-specific features to improve the classifier. Finally, ablation studies can be performed to better understand the impact of different features in this task.

References

1. Ian Soboroff, Shudong Huang, and Donna K. Harman. Trec 2018 news track overview. 2019.
2. Kuang Lu and Hui Fang. Paragraph as lead-finding background documents for news articles. In *TREC*, 2018.
3. Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

4. Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2008.