ICTNET at TREC 2019 News Track

Yuyang Ding[1,2], $XiaoyingLian^{1,2}$, $HouquanZhou^{1,2}$, $ZhaogeLiu^{1,2}$, $HanxingDing^{1,2}$, $ZhongniHou^{1,2}$
1.University of Chinese Academy of Sciences, Beijing, China
2.CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology
{DingYuyang18s,LianXiaoying18s,ZhouHouquan18s,}@ict.ac.cn
{LiuZhaoge18s,DingHanxing18s,HouZhongni18z}@ict.ac.cn

Abstract

This paper describes our work in the background linking task and entity ranking task in TREC 2018 News Track. We explore four methods in background linking task and two methods in entity ranking task. All of our methods largely rely on off-the-shell open-source components(e.g., Solr for indexing the documents), and follow the basic thoughts—BM25 and relevance feedback. These methods differ in how they analyze the given input to obtain a query and to what extent the returned results are re-ranked taking meta data of the document(e.g., publish dates) into account.

1. Introduction

The News track focuses on information retrieval in the service of helping people read the news. In this TREC, there are two tasks–background linking and entity ranking, containing 600,000 news articles aiming to find how news is presented on the web.

The goal of the background linking task is to develop evaluation data to support researchers in developing systems that can help users contextualize news articles as they are reading them. For example, a user is reading a specific article (the query article), algorithms should recommend articles that this person should read next that are the most useful for providing context and background for the query article. It is reasonable to view this task as a specific kind of news recommendation task that would be useful in any news reading context, including the Posts website.

The second task, entity ranking, aims at separating important entities from non-important ones within an article. In addition to providing links to articles that give the reader background or contextual information, journalists sometimes link mentions of concepts, artifacts, and entities to internal or external pages with in depth information that will help the reader better understand the article. Given a news article with title and content that is annotated with entity links to a set of entities, we are supposed to rank the the given entities by importance for the article (i.e., saliency).

To solve this two tasks, we follow the basic thoughts in information retrieval systems—relevance feedback and BM25. Also, we support information retrieval research using the popular open-source Solr search library which allows researchers to easily replicate results with modern ranking models on diverse test collections. These methods turn out to perform well. In our participation at the news track, we submitted four different runs: three for background linking and the last for entity ranking.

The rest of this paper is structured as follows. Section 2 and 3 present the basic frameworks of the two news tasks: background linking and entity ranking. Each section contains several subsections explain the data analysis, related methods, results and the final analysis. Section 4 concludes the paper.

2. Background Linking

2.1. Kicker

As mentioned in Guideline, articles from the "Opinion", "Letters to the Editor", or "The Post's View" sections labeled in the"kicker" field, are not relevant. In order not to miss information, we observe the distribution of documents according to the "kicker" filed. The results of the observation are shown in Figures 1.

There are a total of 184 different types of "kickers". Documents without "kicker" filed are also counted as one category named "None". The "kicker" field can recall up to 39,885 documents and 3,323 documents on average. The median number of recalled documents is 1585. The number of documents that can be recalled by the three types of kickers that need to be ignored is 23074.

Figure 1 shows the distribution of documents recalled by the kicker field. The horizontal axis is the number of recalled documents, and the vertical axis is the number of "kicker" types. There are 22 types of "kicker" recalled documents below 10, and 90 types of "kicker" recalled between 1000 and 10,000. It can be seen that the number of docu-

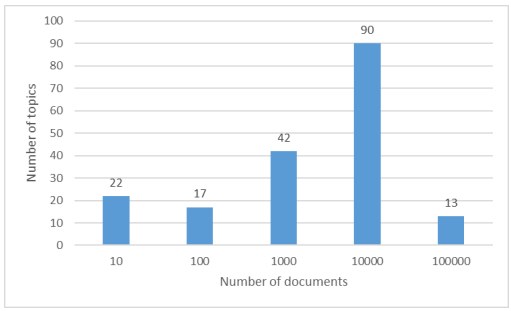

Figure 1. The distribution of documents recalled by the kicker field. The horizontal axis is the number of recalled documents, and the vertical axis is the number of "kicker" types.

ments recalled by most "kicker" is huge, so the types of documents recalled by same "kicker" can be very different. Since there are many differences in documents belonging to the same "kicker", and documents belonging to different "kicker" also have many similar documents. Therefore, we believe that the information brought by the "kicker" field is too confusing to help retrieve the relevant documents. In the following work, the information of "kicker" field is abandoned.

2.2. Methods

Background linking is similar to ad-hoc search task. We use the main article of one topic as query to create a candidate set of background documents with high retrieve scores. Unfortunately, we don't have enough labelled data. Therefore, the keys to retrieve better background documents are constructing a suitable query clause and making good use of output documents with high confidence from unsupervised method. Based on these motivations, here are our methods.

2.2.1 BM25 Baseline

BM25 [5] is one of the most popular and also the best models in the history of information retrieval. We use the default implementation of BM25 in solr to make our experiments. Different from ad-hoc search task with a short query as input, an article as query could have two defects. Firstly, the influence of some relevant words maybe impaired by a large set of irrelevant words. Secondly, the weights of the paragraphs and titles of articles should be set differently to simulate the attention of human readers.

As a consequence, we try a lot combinations of paragraphs and titles to make better query clause: only title, only first paragraph, title+first paragraph, concatenation of every paragraphs, first sentences, all paragraphs. We find that the best one was still the concatenation of all paragraphs. We also search optimal weights combinations and finally set the weight of content as 0.7 and the weight of title as 0.3.

The other methods are developed from this baseline model.

2.2.2 BM25 with Rocchio

Except the dimension of query clause, we can also improve performance by making use of results of unsupervised method, which is also called pseudo relevance feedback. The main idea of relevance feedback is that a user can't fully express his intention on the first try. So we need to expand the inaccurate query to its accurate semantic meaning by user feedback. Specially, we don't have labelled feedback. So we need to do pseudo relevance feedback by the high-confidence subset of output documents from BM25.

In this method, we use the well-known Rocchio algorithm [2]. Rocchio algorithm is a combination of Vector Space Model(VSM) and pseudo relevance feedback model. We see query articles and candidate articles as tf-idf feature vectors. Then the task of retrieve is converted into calculating the similarity between two vectors. As figure 1, the distribution of feature vectors in space of relevant documents is different from irrelevant documents. So we can use them to modify the representation of query articles.

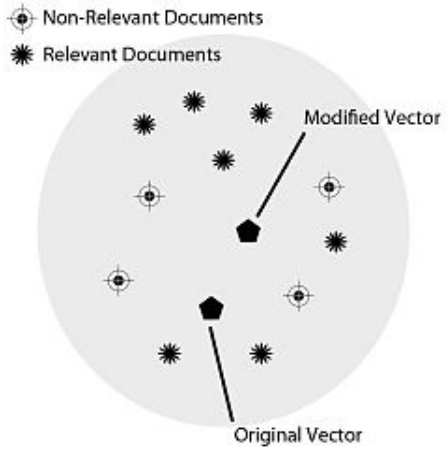

Figure 2. Rocchio algorithm

The formula for Rocchio relevance feedback is as follows:

$$\vec{q}_{opt} = (1-\alpha-\beta)\vec{q_{org}} + \alpha\frac{1}{|C_r|}\sum_{\vec{d}_j \in C_r}\vec{d}_j - \beta\frac{1}{|C_{ir}|}\sum_{\vec{d}_j \in C_{ir}}\vec{d}_j \quad (1)$$

We use BM25 to produce top 100 candidate documents and select the top 20 relevant documents as C_r and bottom 20 irrelevant documents as C_{ir} to create a new query which is closer to the relevant and apart from the irrelevant. In the last, we use the new query to retrieve top 20 documents by BM25.

2.2.3 BM25 with Bert

Another way to make use of BM25 is to learn its pattern of rank of candidate documents. Although there are no labelled relevant document list data, we think it still meaningful to apply learning to rank method on high-confidence output of BM25.

Bert [1] is the state of art neural network model which is applied to almost all fields in NLP. So we apply pretrained Bert model on the top 10 documents from BM25. Specifically, we use some tricks to improve the result. First we select the first paragraph of query and then randomly select some other paragraphs to create a query. We do the same thing to create a candidate document. Then Bert model learns the score of the concatenation of the query and candidate. Second according to the rule of pair-wise learning to rank, we also create some "weak queries" which not include the most important first paragraph but include the other paragraphs and then pair them with the former "strong queries". Bert model learns from the pair and tries to make the weak ones score lower than the strong ones. These tricks help to create more data and build a more robust model.

2.2.4 Query likelihood and Relevance Model 3

Query likelihood(QL) [4] is a well-know information retrieval method. Based on language models built from each document, QL ranks documents according to the likelihood of the given query.

QL is usually combined with relevance feedback approaches, especially RM3. RM3 is an variant of relevance-based language model(usually called RM)[3], and most frequently-used in information retrieval tasks. RM assumes that there exists a relevant language model and builds a language model based on these results. The retrieval is achieved by standard QL language model and this auxiliary language model.

$$P_{rm1}(t \mid q) \propto \sum_{D \in D_R} P(t \mid t) \prod_{q_i \in q} P(q_i \mid D) \tag{2}$$

$$P_{rm3}(t \mid q) = (1 - \lambda) \cdot P_{MLE}(t \mid q) + \lambda \cdot P_{RM1}(t \mid q) \tag{3}$$

We test the standard QL model and QL+RM3 model. In the former, we use Jelinek-Mercer smoothing method in language model and apply the parameter λ to 0.7 for queries are long. In the latter, we use top-20 results returned by QL as relevance documents and apply the rm3 interpolation parameter to 0.4. We test constructing query in different ways. Since the title and the first paragraph usually contains useful information about the news, we try constructing query with title and title+first_paragraph respectively. Furthermore, we make the full text as query as well.

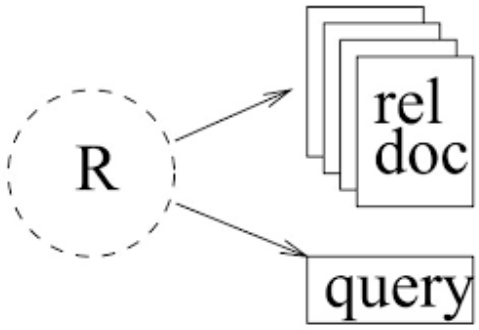

Figure 3. Relevance Language Model 3(RM3) algorithm

	BM25	Rocchio	QL	RM3
nDCG@5	0.5757	0.4438	0.5651	0.5633
MAP	0.389	0.3157	0.3948	0.4012
Rprec	0.4446	0.3739	0.4545	0.4506

Table 1. Results for all topics (median) and our four methods on the background linking task

2.3. Result

We report results of our runs in the background linking task in Table 1. We observer that the BM25 baseline run performs best on nDCG@5 in total 60 topics, whereas the RM3 and Query likelihood respectively perform best on MAP and RPEC.

The precision-recall curves in Figure 4 shows that the performance of RM3, Query likelihood and BM25 are nearly similar, and the Rocchio method where the pseudo-relevance feedback is used based on BM25 significantly performs worse than the other methods. This indicates that the news takes a deep and complex knowledge relevance with its related background news. As a result of this, simply expanding query with similar semantic words cannot improve the performance of the query model.

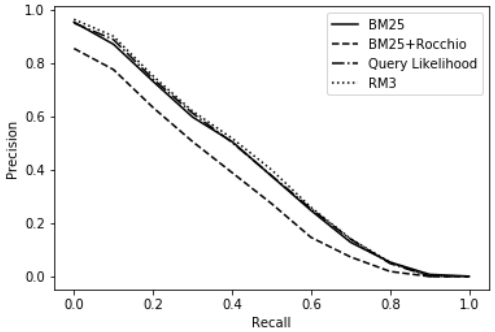

Figure 4. Precision-Recall Curves for the four methods

To further investigate the performance of our methods

Method	Gain/Loss	Topics
BM25	+	880, 841, 874
BM25	-	855, 829
Rocchio	+	880, 841, 882, 884
Rocchio	-	855, 867,858,830
Query Likelihood	+	880, 885, 841, 874
Query Likelihood	-	867, 858, 829
RM3	+	880, 885, 841, 861, 874
RM3	-	867, 858, 853

Table 2. Topics for which our methods achieved the highest gain (+) or loss (-) on the background linking track

and explain how pseudo-relevance feedback degrade performs of BM25, cases analysis was performed on the result for each topic. We list the topics where our methods gain highest and lowest nDCG@5 scores in Table 2. It has been observed that Rocchio method where expand query for BM25 gains less score than the original BM25 on the topic 874. After reading the title and content of the article in the topic 874, we find that some metaphor are used in the title to describe the main idea of the article, i.e., the phrase "house flipping" to describe a housing exchanging. It is clear that the semantic of the word "flipping" has been changed greatly from its original definition when it combines with the word "house". However, when pseudo-relevance feedback is used to expand the query which is based on the article title, it has large probability to add synonyms whose semantic is same as the original definition of the word "flipping" to the query. Hence, more articles which contains the word with its original semantic are retrieved to the results, and the performance of the Rocchio is definitely worse than the method without pseudo-relevance feedback.

3. Entity Linking

3.1. Method 1

Entity linking task aims to sort entities in a news article according to their salience. Each entity has a wiki id. In this method, we relate each entity with the corresponding wiki page and regard the entity linking task as retrieval in these wiki pages. The returned ranking is used as the entity ranking.

Given the official wikidump data, we first build index on wiki pages of entities appeared in the dataset. It is efficient since we do not need to build index on all wiki pages. We consider the news article as "query" and wiki pages of entities as documents to be retrieved. Similar to background linking task, we try different query constructing methods including title, title+first paragraph and the full text. We use BM25 as our retrieval model and the parameters are tuned.

	M1	M2
nDCG@5	0.6800	0.7191

Table 3. nDCG@5 for all topics (median) and our two methods (M1, M2) on the news track 2018 dataset

The retrieval results are filtered to make sure they only contains entities occured in this news article.

3.2. Method 2

Method 2(M2) and Method 1(M1) have the same central idea. They all think that the more related to the query document the wiki page is, the more important is the entity in the query document which owns the wiki page. That is, if the wiki page owned by the entity is ranked higher in the result set returned by the query document, then the importance of the entity is higher, and the final score should be larger. The correspondence between the entity and the wiki page is determined by the "enwiki" owned by each wiki page, and a wiki page will contain multiple "enwiki".

M2 assumes that if the "enwiki" of an entity e_i is in the "enwiki" of a wiki page W_i, then considers that the wiki page W_i corresponds to the entity e_i. That is, the entity e_i owns the wiki page W_i, and the wiki page W_i also points to the entity e_i. The biggest difference between M2 and M1 is that each entity e_i selects multiple related wiki pages W_i to build an index. In a query, the final ranking $rank_i$ of each entity e_i is only related to the ranking of the highest ranked wiki page W_{ibest} in the query result set.

The specific process of M2 is as follows. First, the top M wiki pages of each entity e_i are retrieved according to "enwiki", and the full-text index of the body of all wiki pages is constructed. At the same time, each wiki page W_i uniquely points to an entity e_i. The entire body concatenate with title of each document is used as the query. Then retrieve N related wiki pages in the full-text index constructed in the previous step as the result set R. Find the $rank_i$ of the wiki page W_{ibest} in the result set R of the M wiki pages owned by each entity e_i. Finally, the entity e_i scores $M - rank_i$ in the document.

The pre-processing of Washington Post and Wiki dumps in M2 only includes uppercase to lowercase, eliminating stop words and stemming. By extending the wiki pages owned by each entity, M2 have become our best way to perform on entity ranking tasks.

3.3. Result

Since at the time of writing, entity ranking runs for other participants are not yet available, we observed the performance of each method on the news track 2018 dataset, using nDCG@5 as the primary effectiveness measure.

The performance of each method is shown in Table 3. As can be seen from the table, M2 has significantly im-

Method	Gain/Loss	Topics
M1	-	810, 818
M1	+	433, 802, 805, 813, 816
M2	-	362, 810, 818
M2	+	347, 433, 802, 804, 806, 819

Table 4. Topics for which our methods achieved the highest gain (+) or loss (-) on the entity linking track

proved the performance compared with M1. M2 mainly benefits from Pseudo relevance feedback, all the wiki pages are point to one entity, which intangibly enriches the information carried by each entity, so that the query can better match the most relevant aspects of the entities, thereby more accurately measuring the entity importance in the document.

To further observe how our method performs, we list some of the highest and lowest nDCG@5 scores in Table 4. M1 and M2 performed poorly on both the 810 and 818 topics, and performed well on both the 433 and 802 topics. It has been observed that it is easy to obtain low scores when there is a majority of unrelated entities in a document, and it is easier to obtain high scores when most related entities are available. This is mainly because our method use rank to distinguish whether the entity and the topic are strongly or weakly related. An entity will always have its final rank whether it is related or not. As a result, our approach tends to think that entities are related to topic, so it performs well on topics with many related entities, but performs poorly on topics with few related entities.

4. Conclusion

In this year's News Track, we take part in all of the two tasks. In the task of background linking, we investigate using paragraphs in different ways to construct our query. We use BM25 to retrieve document and combine Rocchio, Bert and other methods to enhance its performance. Results show that the performance on long text is unsatisfactory.

In the task of entity ranking, we follow the basic idea that the more related to the query document the wiki page is, the more important is the entity in the query document which owns the wiki page. We select multiple related wiki pages for each entity to build an index. It perform well on topics with many related entities while perform poorly on topics with few related entities.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[3] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 120–127, New York, NY, USA, 2001. ACM.

[4] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[5] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.