NTUBROWS System for NTCIR-7 Information Retrieval for Question Answering

I-Chien Liu, Lun-Wei Ku, *Kuang-hua Chen, and Hsin-Hsi Chen

Department of Computer Science and Information Engineering,

*Department of Library and Information Science

National Taiwan University

Taipei, Taiwan

{icliu, lwku}@nlg.csie.ntu.edu.tw; khchen@ntu.edu.tw; hhchen@csie.ntu.edu.tw

Abstract

This paper presents an information retrieval system for the NTCIR-7 information retrieval for question answering task. This system is composed by three parts: (1) Query processing (2) Retrieval model (3) Re-rank module. Query processing filters stop-words and selects query terms to generate a required term set for further retrieval. Threes retrieval models from two famous retrieval systems are adopted. They retrieved relevant documents based on the generated required set. Finally, Re-rank module gives documents scores according to the distributions of their possessive terms. The performance of our system achieves a mean average precision (MAP) of 0.4635, a Q-measure of 0.4811, and an MSn-DCG of 0.6831 on NTCIR-7 IR4QA testing set.

1. Introduction

Queries present users' information needs. Traditional information retrieval systems compute a numeric score on how relevant each document is to the query, and rank the documents according to this score. However, as technology improves, users may wish to find relevant documents by querying with natural questions instead of terms or some predefined topics.

The aim of information retrieval for question answering is different from original information retrieval. It is to see how the performance is when the query is a question, or at least question-like. It may give a reference for how is the performance before answering questions, which should be the upper bound for the performance of question answering. However, with this task description, we may also design algorithms using some attributes of questions, or discuss results by question types.

For this task, we borrowed some ideas traditional QA task and developed a three-phrased system. The first phrase is query processing. Query processing performed a special stop-word filtering and produced a required set of query terms. The concept of stop-words filtering is common in natural language processing, but we define stop-words by analyzing queries (questions). The second phase, applying retrieval models then utilized this required set to retrieve relevant documents. The concept of "required" which treats the selected terms as in the boolean model is also inherent from techniques in QA domain. In the last phase, a re-rank module was applied to reorder the result list. This re-rank model considers paragraph-level information and hence more similar to the relevant passage retrieval in QA task.

The remainder of the paper is organized as follows. Section 2 contains an overview of our system and describes the methods to implement it. Section 3 includes the evaluation results. Section 4 is a discussion on error analysis. Finally, section 5 concludes this paper and indi-

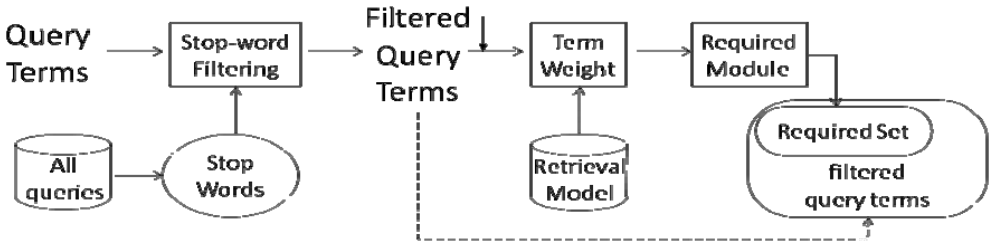

Figure 2. Query processing workflow.

cates the direction of future work.

2. System Description

Figure 1 shows the framework of our IR4QA system. In this system, we first chunk a query into several terms, which are then passed to the retrieval model. Retrieval model outputs a list of relevant documents in order. Finally, these records reported by the retrieval model are re-ranked by the re-rank module.

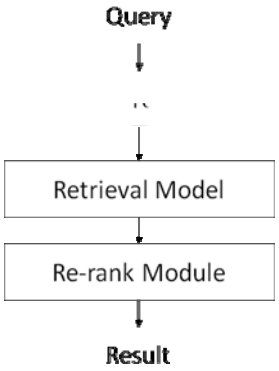

Figure 1. System framework.

2.1. Query processing

To select query terms, we only use the words in the "QUESTION" field, since the "NARRATIVE" part does not seem to provide more information at the first sight. Then we try to remove stop words from these query terms. These sentences in the "QUESTION" fields are all questions, and their possessive terms play different roles. What we try to remove are those terms which appear frequently in questions, and they are defined as stop words in this paper. Therefore, instead of look up in a general stop-word list, we count the frequencies of each term in all queries and drop it as stop words if the frequency is over 20. Query processing workflow is shown in Figure 2.

Except those terms we may not need in a query, the importance of other terms is not all equal. A “required set” here denotes a set of terms which is considered more important. The required set try collect discriminative terms. To select terms into the required set, we first calculate their weights. The weight of term t is calculated as follows:

$$weight(t) = \sqrt{\sum_{d \in D(t)} score(d,t)^2} \qquad (1)$$

where D(t) is the set of documents containing term t, and score(t,d) is given by the retrieval model (will be introduced in Section 2.2). The score(t,d) represents the score of term t in specific document d. Terms will be marked as required when it’s weighted higher than the threshold. There is one threshold for each query and it is set to 70% of the maximum weight among terms in each query.

$$RS_q = \{t \in q \mid weight(t) > MAXW(q) * 0.7\} \qquad (2)$$

Where $MAXW(q) = \max_{t \in q} weight(t)$

2.2. Retrieval model

Unlike words in the Latin language, Chinese words don't have obvious boundary divided by spaces. Chinese word segmentation is usually necessary for determining the word boundary before processing Chinese texts. A sentence is considered as a serial of characters before segmentation. After segmentation a sentence can be considered as a serial of words. Therefore, there are two different types of information retrieval models, word-based model and character-based model. The basic unit to be indexed in a word-based and a char-based retrieval model is a word and a character, respectively.

Here we demonstrate the differences between two models by an example. The sentence "我 今 天 穿 牛 仔 褲" (I wear jeans today) is a sequence of seven characters. After segmentation, it becomes a sequence of four words "我 "(I), "今 天"(today), "穿"(wear), "牛 仔 褲 "(jeans). If we retrieve by the query term 牛仔

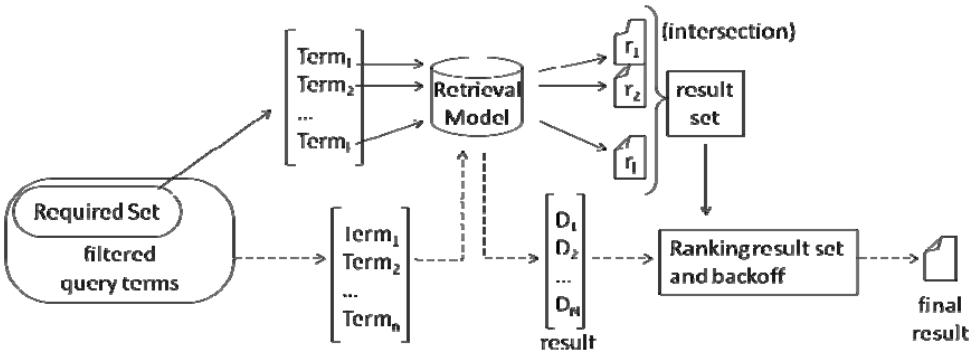

Figure 3. Retrieval model with required set.

(cowboy, one word composite of two characters), this sentence will be retrieved in the char-based model but not in the word-based model, since it is only partially matched with the word "牛仔褲" (jeans). Though it seems better to adopt a word-based model in this case, the performance of the word-based retrieval model may be influenced by the performance of the word segmentation.

We adopt models of two well-known information retrieval systems, Okapi and Lucene. They were chosen because their basic units of indexing are different. Okapi is a word-based retrieval model. Any words need to be exactly the same as indexed if they can be retrieved. Lucene provides both a character-based model and a special word-based model which retrieves documents in which words match partially within one word with the query term, but words partially match the query term cross the boundary of words will not be considered relevant.

Figure 3 illustrates how the retrieval system works. As mentioned in Section 2.1, a required set is generated for each query. For each term in the required set, we used it to query Okapi/Lucene and got a result list of relevant documents. We then intersect these records and generate a new result set. After intersection, every document in the result set is guaranteed to have all terms in the required set. In other words, it performs the "AND" operation on search engines.

At last, all query terms are used to query Okapi/Lucene together, and a list of relevant documents, D_1, ... , D_N is obtained. For each document D_i, we keep it as in the final result set if D_i is in the current result set. This step gives ranks to the documents in the current result set. However, if the size of the current result set is less than 1000, we will not have a final result set containing 1000 documents. In this case, we selected documents to append it to the final result set from those haven't been selected among D_1 to D_n by their ranks. For instance, if the current result set only contains D_2 and D_3, the final result will be D_2, D_3, D_1, D_4, D_5, ... , D_{1000}. In this manner, we can keep the ranking order given by Okapi/Lucene for documents in the current result set but still assign them higher ranks.

2.3. Re-rank module

After we generate the final result set and decided their ranks, a re-rank module is adopted to adjust the rankings considering only these relevant documents. We adopt another weighting formula to re-weight terms in these documents [3]. The formula re-scores terms by taking many different features in consideration, such as frequency, position in paragraph, term's distribution, etc.. Using this formula, we can give documents new scores by combining scores of these document terms. Since the formula takes term distribution in consideration, the re-rank depth should be limited. Otherwise irrelevant noises may decrease the performance. In official runs a re-rank depth 200 is applied.

Another issue is that the re-rank module assumes its input documents are mainly discussing the same topic. In fact, this assumption really harms performance in this application and we will discuss the reason in Section 4.4.

3. Experimental Results

The official corpus should be CIRB-020 and CIRB-040. By mistake, we included CIRB-011 as our experimental corpus, too. Therefore, lots of documents that are not included in the official dataset are retrieved in our official runs and evaluated as negative answers.

The metrics to evaluate the performance of this task is Mean Average Precision (MAP), Q-measure, and MSn-DCG. All of these me

Run ID	Filtered			Original		
	AP	Q-measure	MSn-DCG	AP	Q-measure	MSn-DCG
NTUBROWS-Run1	***0.4628***	***0.4795***	***0.6711***	0.3587	0.3780	0.6400
NTUBROWS-Run2	0.2300	0.2772	0.5239	0.2008	0.2498	0.4993
NTUBROWS-Run3	0.2359	0.2788	0.5090	0.2129	0.2495	0.4853
NTUBROWS-Run4	0.2371	0.2811	0.5158	0.1935	0.2303	0.4640
NTUBROWS-Run5	0.1871	0.2218	0.4208	0.1653	0.2026	0.4041

Table 1. Original and filtered results of five runs.

Model	NTUBROWS-Run	AP	Q-measure	MSn-DCG
Lucene_Char_Baseline_Filtered		0.3672	0.3881	0.5877
Lucene_Char_Baseline_Filtered_Rerank		0.2091	0.2476	0.4743
Lucene_Char_Required_Filtered		0.2597	0.2796	0.4800
Lucene_Char_Required_Filtered_Rerank	Run5	0.1871	0.2218	0.4208
Lucene_Word_Baseline_Filtered		0.4457	0.4669	***0.6831***
Lucene_Word_Baseline_Filtered_Rerank		0.2425	0.2897	0.5384
Lucene_Word_Required_Filtered		0.4353	0.4543	0.6725
Lucene_Word_Required_Filtered_Rerank	Run2	0.2300	0.2772	0.5239
Okapi_Baseline_Filtered		***0.4635***	***0.4811***	0.6760
Okapi_Baseline_Filtered_Rerank	Run4	0.2371	0.2811	0.5158
Okapi_Required_Filtered	Run1	0.4628	0.4795	0.6711
Okapi_Required_Filtered_Rerank.xml	Run3	0.2359	0.2788	0.5090

Table 2. Results of all the combination of our system.

trics are based on the whole list of documents returned by the system and reword earlier return of true positive records. Their specifications are described in overview paper [4].

Five runs are submitted officially by us. Their settings are shown in Table 3.

NTUBROWS-RUN	1	2	3	4	5
Okapi/Lucene	O	L	O	O	L
Char/Word	C	W	C	C	C
Baseline/Required	R	R	R	B	R
Re-rank (N/Y)	N	Y	Y	Y	Y

Table 3. Settings of five official runs.

Table 1 shows the original results and the results of filtering out CIRB010 answers. AP and Q-measure are improved about 19% and 17%, and MSn-DCG is improved nearly 6% for each run. The best result is Run1, which achieved a MAP of 0.4628, Q-measure of 0.4795, and MSn-DCG of 0.6711.

Table 2 shows the experimental results of all the combinations we have, i.e., Okapi or Lucene, using baseline or required query, and re-ranked or not. We use the following naming principal.

- Okapi/Lucene: Beginning with the term Okapi or Lucene indicates which base retrieval model was used.
- Char/Word: There are character-based and word-based models in Lucene, but Okapi is word-based.
- Baseline/Required: Models named baseline are not using the required set described in Section 2.2, but using the original query terms with stop words removed.
- Filtered: The result list does not include documents in CIRB010.
- Rerank: Applying re-rank module to the result set.

The best setting for our system is using Okapi without re-ranking, which achieved an aver-

age precision 0.4635 and a Q-measure 0.4811. However, for the MSn-DCG score, Word-based Lucene without re-ranking is the best setting, which achieves an MSn-DCG score 0.6831.

4. Discussion

After analyzing experimental results and datasets, we find several issues that decreasing the performance. They are discussed in the following sections.

4.1. Dataset

As mentioned in Section 3, we mixed CIRB-011 into the corpus which is not in the official dataset. We retrieved about 17,000 documents from CIRB-011 in each run, and they were all evaluated as negative records. It's a drag on our performance. After filtering out those documents, the MAP and Q-measure are improved over 19% and 17%, and MSn-DCG is improved nearly 6%. Table 1 shows the results before and after filtering out the answers from wrong dataset.

4.2. Quality of query terms

In traditional information retrieval tasks, we have several sources for query terms like titles, descriptions, narratives, and concepts etc.. Having these sources, we can generate much more query terms for the retrieval model. In this task, the "QUESTION" field of the official topic file is the only source for us to get the information needs. Other fields seemed not to provide more information. Lack of query terms lead to a poor performance of our information retrieval system, and also it is a major problem to be solved in the future.

4.3. Results of retrieving relevant and partial relevant documents

From experimental results, we found that submitted Run2 and Run3 have less mean average precision but higher MSn-DCG scores. Because MAP considers the amount of correct answers while MSn-DCG calculates different gains from reported "relevant" and "partially relevant" documents, we suspect that our system is weak in retrieving partially relevant documents. To confirm this postulation, we calculate the percentage of retrieved "relevant" and "partially relevant" documents in our results. We then find that the number of retrieved "partially relevant" documents is about 4% to 7% fewer than that of "relevant" documents. This result suggests that we may need to relax the restriction when retrieving documents or find more clues to retrieve more partial relevant documents.

4.4. Pre-conditions for Re-ranking

Table 2 shows that the re-rank module did harm to the system performance. After re-ranking, the performance of each model drops around 45% compared to the original. After analyzing system performance by query, we find that the re-rank module improves the performance only when the original retrieved documents are of a certain quality. In other words, a certain percentage of the retrieved documents before re-ranking must be relevant or at least partial relevant. Therefore we conclude that if the quality of original result is ensured, re-rank module will help. If not, it is not suitable to apply re-rank module.

For example, Topic id ACLIA1-CT-T200 is the topic concerning "Who is Pope John Paul II?" On this topic, we retrieved 13 relevant results and 87 partial relevant results in top 200 documents, which is our re-rank depth. In this case, half of the documents being re-ranked are relevant to the topic and the MAP improves from 0.2413 to 0.3943.

But there are some other topics which don't have more than 50 relevant documents in top 200. In such topics, the re-rank module will not help because irrelevant documents involved are way too much than relevant documents. In this case, non-relevant subtopics in the document set will be weighted more than relevant subtopics so that irrelevant documents will be ranked higher, because irrelevant information overwhelms the relevant.

4.5. Topic related issues

We find that several issues which influence the performance are related to topics. In this section, we will discuss three issues of them.

First, evaluation results of topics of different query (question) types vary. There are four query types in IR4QA task, i.e., event, definition, biography and relationship. Topics of query type relationship have worst performance in our experiments. These topics ask about the relationship between some entities A and B. If one of A and B appears much more frequently in the document set than the other, the retrieved documents will be dominated by it and hence not

relevant to the relationship. Thus, the performance was not satisfactory.

Second, topics concerning conditioned numerical range perform badly. Some topics like "since 1997" or "no less than 6.8" are not easy to deal with since our system does not involve the pre-processing of queries like in the traditional question answering tasks. Therefore, the system was not able to regard the numerical constraints for these topics and led to a bad performance.

The third issue concerns abbreviations in topics. For example, "以色列-巴基斯坦"(Israeli-Palestinian) is abbreviated to "以巴" (I-Pa). This decreases performance since the retrieval model does not understand the most important terms, these abbreviations, in queries. These abbreviations may be segmented wrongly in the first step. Even if they are segmented correctly, there is no way to find them in relevant documents without expansions. As a result, our system achieved relatively low performance for topics containing abbreviations.

5. Conclusion and Future Work

In this paper, we propose an information retrieval system that achieves satisfactory performance. This system is based on a required set of terms and a re-ranking approach to adapt the system from the traditional IR task to the IR4QA task.

We evaluate the results of all designed experiments. Our re-rank module is effective when the original retrieved documents are mostly relevant, but there are still some cases where the performance is not so satisfactory. These cases involve the selection of query terms and the characteristics of topics.

Since we have found the defects of our system, many methods for improving it could be experimented in the future. For example, we could incorporate a query expansion algorithm into our system to enhance the performance of the retrieve model. We could also construct a query model utilizing the query type to enhance our system and incorporate with the backend QA system.

References

[1] OKAPI information retrieval system http://www.soi.city.ac.uk/andym/OKAPI-PACK

[2] Lucene information retrieval engine http://lucene.apache.org

[3] Ku, L.-W., Lee, L.-Y., Wu, T.-H. and Chen, H.-H. (2005). Major topic detection and its application to opinion summarization. SIGIR 2005, pages 627-628.

[4] Sakai, T., Kando, N., Lin, C.-J., Mitamura, T., Ji, D., Chen, K.-H., Nyberg, E.: Overview of the NTCIR-7 ACLIA IR4QA Task, Proceedings of NTCIR-7, to appear, 2008.