FDUQA on TREC2006 QA Track

Yaqian Zhou, Xiaofeng Yuan, Junkuo Cao, Xuanjing Huang, Lide Wu
Fudan University, Shanghai, China, 200433
{zhouyaqian,xfyuan, jkcao, xjhuang, ldwu }@fudan.edu.cn

1. Introduction

In this year's QA Track, we participant in the main task and do not take part in the ciQA task. The main task is essentially the same as the single task from 2004, in that the test set consists of a set of question series where each series asks for information regarding a particular target. In order to better answer the questions in the series, we try to improve our anaphora resolution within question series.

For factoid questions, we use the system that submits the RUN-A in TREC 2005[Wu et al. 2005]. Therefore we won't describe the factoid system in this paper.

For list questions, we get a lot of improvements, the most important of which are answer type classification, document searching, answer ranking and answer filtering.

For definition question, we still focus on utilizing the existing definitions in the Web knowledge bases. And also applied the method of relative terms extraction to extract reliable information associated with target for getting web definition directly by question target is becoming a bottleneck.

In the following, Section 2 will describe question series anaphora resolution. Section 3,and 4 will describe our algorithms for list and definition questions separately. Section 5 will present our results in TREC 2006.

2. Question Series Anaphora Resolution

Because the test set consists of a set of question series where each series asks for information regarding a particular target, it is necessary to do anaphora resolution.

There are three kinds of anaphora in the question series: coreference, bridging and zero anaphora. In our system we only resolve the coreference. And considering target as additional information when the question sentence has the other two kinds of anaphora..

In order to resolve coreference, we need to find anaphor and antecedent at first. There are three types of anaphors: the pronouns, the dBNP (Definite Base Noun Phrases) and the abbreviations. But not all the dBNP are anaphors, there are some exceptions: the answer of the question (e.g. *the name* in "*What were the names of the victims?*"), some special words (e.g. *earth*, *sky*, etc.). The candidate antecedent may be in the target, in previous questions, or in the answer of the previous questions.

We find that the syntax relation between anaphor and antecedent is not very close. It is difficult to resolve coreference according to the syntax restriction. Therefore we resolve coreference according to the coherent of type, gender and number between the anaphor and the candidate antecedent. We divide the type of anaphors and antecedents into six categories:

person, organization, location, event, time and other.

During the resolution, we first classify the anaphor. Then the candidate antecedent has the same type is selected as the anaphor. The priority of selecting candidate antecedent is according to their position:

target >answer of previous question (nearer question>further question)> the noun phrase in previous question (nearer question>further question).

If the type, gender and number are all matched, the resolution is finished. If the antecedent of some anaphora has appeared in the current questions, the resolution will not be done, or the result of resolution will be not in the habit. If there are not any coherent candidate antecedents, the resolution will not be done.

3. List Question

Our list question answering system adopts the general factoid question answering framework and composes of three modules, answer type dassification module, passage retrieval module, and answer generation module.

The answer type classification module considers the character of list questions. The passage retrieval module focuses on the recall of the passage returned, which is more important in list questions. And the answer generation module gives a strategy of how to extract and select multiple answers from different documents.

3.1 Answer Type Classification

We first create a two-level ontology to define the answer type. It has six coarse-grained answer types and ninety-seven fine-grained answer types. And each fine-grained answer type is mapped to a coarse-grained answer type. Then, we classify the question into one of the fine-grained answer type based on WordNet. We start from the informer of the question, then use following algorithm to find its answer type:

```
Find the informer IM of the question in WordNet
For each sense S of the IM
Do
    W←S
    While(W is not NIL)
    Do
        If (W is in any of the fine answer types)
            Output W
            Exit;
        End if
        Else
            W←the Hypernym of W
        End else
    End while
End for
```

Output NIL

After we get the output of the fine-grained answer type, we can immediately get the coarse-grained answer type from a type mapping.

Testing on the 93 questions of TREC 2005, the above simple algorithm can get a satisfying precision 87% on the fine-grained answer type classification.

The informer of a question is a word or a phrase that helps to classify the type of the answer. For example, given a question "What book did he wrote?" the informer of the question is the word "book". It is very useful for the classification of the answer type of a given question. [Krishnan et al, 2005] also find out the informer is a key element for classifying a question's answer type.

Compared with [Krishnan et al, 2005], we use a more semantic and easier way which also has a high precision in the list questions to extract the informers in a question. Due to the specialty of the List questions which usually start with some definite words such as what, list, name, etc., we use a relation parser Minipar to extract the informers which are related to these specific words of list questions. Minipar can extract such a triple: (K R S), where K represents the informers we need, S is the specific word in the question, and R is the relation between the informers and the specific words. As of the example above, Minipar will output such a triple: (book det what), thus the word "book" is extracted as the informers of the question, *what* is used here as the specific words and their relation is *det*. We use some heuristic rules to choose the specific words from the question and the relations from the minipar. And the precision of extracting informers of the 93 TREC 2005 list questions is 95% (5 are wrong).

3.2 Passage Retrieval

3.2.1 Document Retrieval

We discover that the document searching strategy in list question answering is quite different from that of the factoid. And some effective document searching methods of factoid question answering are not suitable for list question answering. There are two main elements in the searching phase of the list question answering, the first factor is the relevance of the target and the question, and the second one is the number of the documents returned. For factoid question, the answer is unique, so we only want the most relevant documents, and the number of documents returned is usually less than 10. While for list question, the number of answers is unknown, so any document that is related to the target or the question may contain the answer, and we should not risk missing any of them in the document searching phase. The number of documents returned is usually more than 50. In a word, list question answering emphasis more on recall than precision. We use a framework, Lucene, as our document search engine. And the result shows our document searching strategy raised the performance by 8% in TREC 2005 list question set and 15% in TREC 2004 list question set (Top 10 document recall). Our list document searching strategy is presented in the following respects: Indexing, query generation and document scoring.

Since we only use the TREC Aquaint as the searching corpus, and no other external

documents, the index process is necessary and important for our system. The main difference between the index for list question and the index for factoid question is that the index is based on the morphed Aquaint corpus. Every word except the proper nouns is morphed into its original form before it is indexed. This index strategy can avoid missing some important documents in some degree.

To match the indexed word, the query words are also morphed. The query is composed of two parts: the target part and the question part. The weight rate of the target and question is 6:4. Also there are two fields in each part: the text field and the headline field. The weight rate of the two fields is 50:1

Lucene is a VSM model based search engine framework, thus the document ranking is based on the basic tf idf score of each dimension of the vector.

3.2.2 Sentence Scoring

Although we get the ranked documents, they may be too long for processing. We wish to find the most important and relevant part of the document. This helps to locate the answer in a more precise range (a sentence). Also the sentence score can compensate for the shortcoming of the tf idf ranking strategy. The very heuristic rules of the sentence scoring process are that the more words of the sentence appear in the target or the question, the higher its score, and that the more important the words of the sentence, the higher its score. According to these two rules, each sentence of each document is given a sentence score ranged from 0 to 1.

3.2.3 Document Re-ranking

In order to get a better ranking result of the documents, we combine the Lucene document ranking score and the best sentence score in a document. Let S be the score of the sentence factor and D represent the score of the document. Our task is to find a function $f(S, D)$ which is the combination of the two scores.

First, we should determine the value of S and D. The score D is simply the similarity returned by the search engine Lucene. While to get the S score, we have several options. Suppose there are N sentences in the document, and each sentence score is S_i ($0<i<N$). Then $S = q(\vec{s})$ is a vector to value function. We've tried the following q functions: Max, Min, Average and Median. Our experiment results show that the Max function is better than any other functions. So let $S = Max(\vec{s})$ in our system.

Second, we will try to find the function f. Only the linear combinations of the two parameters of f are considered. So f is a linear function and has the form $f = a*S+b*D$. a, b are the parameters to be determined. Our experiments show that when a =0.7, b =0.3, the ranking is most improved.

3.3 Answer Generation

3.3.1 Answer Extracting

Although the document is re-ranked, we still need a more precise and small range to extract answer, for the extracting process is time consuming. We extract answer from sentences ranked from 1 to N in the previous stage. Since the context information of each sentence may also contain the right answer, we also extract answers from the next K sentences and previous K sentences of the top ranked sentences. Here, the N is supposed to be 200 and K is 2.

The extraction is based on the Named Entities and some chunked phrases in the sentence. According to the answer type that has been classified, we extract relevant named entities. The answer types used here is the coarse one because our named entities classifier can only classify the coarse types.

During the extraction, a distance score which is associated with each answer is also calculated. This score is used to indicate how far it is from the answer to the important words of target and question. Assuming that there are d words between the candidate answer c and the important word k, the distance score is calculated by $f(c,k) = a \cdot \left(\frac{1}{d+0.5}\right) + b \cdot \left(1 - \frac{1}{w}\right)$, where w is the weight of the key word k in the question or target. a, b is the parameter of the function and $a + b = 1$. $a \cdot \left(\frac{1}{d+0.5}\right)$ is the distance measure of c and k, $b \cdot \left(1 - \frac{1}{w}\right)$ is the importance measure of k. In our experiment we set $a = 0.5, b = 0.5$. The final maximum distance score $S_{\max}(c)$ is the maximum $f(c,k)$ of all k, and the final average distance score $S_{avg}(c)$ is the average $f(c,k)$ of all k. Each extracted candidate answer has a maximum distance score and an average distance score together stored with it.

3.3.2 Answer Ranking

Since for each candidate answer, we've already calculated several scores, the ranking procedure is to sort the candidates according to a combination of the scores. We use the simplest linear combination of three scores; they are the Lucene similarity score, sentence score, and the maximum distance score. So, the final score of a candidate answer is $S = a \cdot S_{Lucene} + b \cdot S_{sentence} + g \cdot S_{distance}$. Our experiment is to determine all the parameters a, b, g. The result indicate that when $a = 0.3, b = 0.5, g = 0.2$, the final F score is best.

3.3.3 Answer Filtering

Although the answers are ranked, there may be some redundancy and noise in the ranked answer list. The answer filtering module is to reduce the redundancy and eliminate as much noise as possible.

To reduce redundancy, we keep the longer answer and delete the shorter one.

Eliminating noise is a difficult job, for we don't know what kind of answer is noise. Thus, we did it only in few cases where the answer type has a finite answer collection (such as country, city, river and etc.). In such cases, a potential answer list for each of these answer types is used to validate the extracted answer, and the answers that do not appear in the list are eliminated. Thus, we partially eliminate the noise, resulting in a high precision.

4. Definition Question

In order to automatically identify definition sentences from a large collection of documents, we first extract related knowledge as much as possible by question target, and then apply the knowledge to pick out the question answers. The knowledge includes online definitions and relative terms. In our system, the extraction of relative terms differs from traditional methods based on calculating the co-occurred frequency of the target words by using the scores of candidate answer sentences for integrative selection.

4.1 System Overview

We design a general architecture for definition QA. The system consists of four modules: document processing, web knowledge acquisition, relative terms extraction and definition generation. The flow chart for definition question of FDUQA is in figure 1.

First, the document processing module generates the candidate sentence set according to the target term. This module has three steps: document retrieval, candidate sentence extraction and initial score calculation.

Second, the Web knowledge acquisition module acquires the definitions of the target term from the Web knowledge base (KB). We search the definitions about the target from a number of online knowledge bases. These knowledge bases are the WordNet glosses and other online dictionaries such as the biography dictionary at www.encyclopedia.com. The definitions from them often supply knowledge that can be exploited directly and they are quite helpful to answering definition questions. We choose several authoritative KBs that cover different kinds of concept to achieve our goal. If we can find the definitions of question target from these sources, we use them to score the candidate sentences. (More detail please refer to [Wu et al, 2004])

Third, we automatically extract relative terms based on the candidate sentences, and then we score the candidate sentences using these relative terms. The extraction of relative terms will be described in Section 4.3.

At last, the definition generation module ranks the candidate sentences, based on target term, relative terms and web knowledge, and determines the question answers. We will

describe the detail of the definition generation module in Section 4.4.

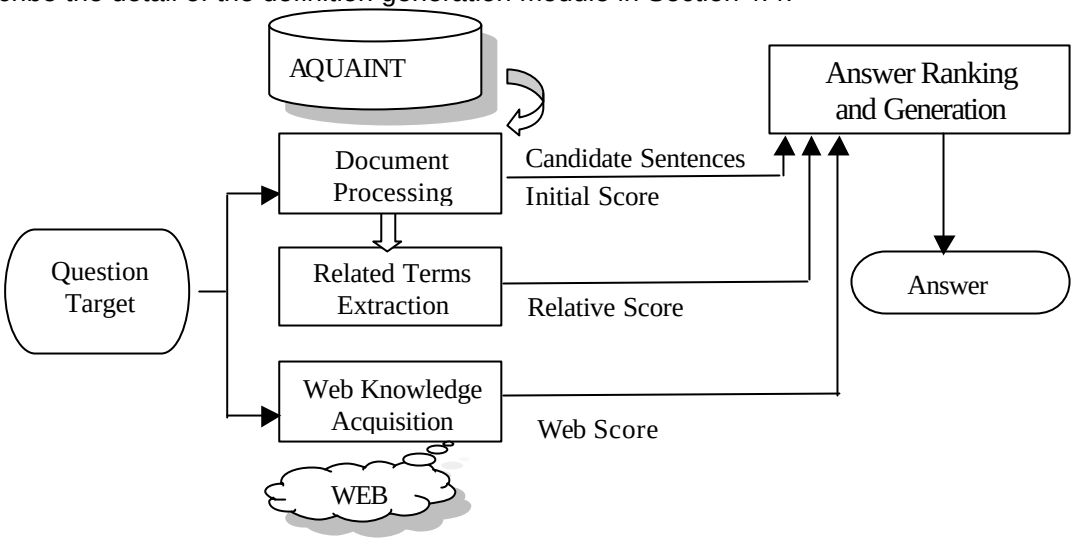

Figure 1: System Architecture

4.2 Document Processing

The document processing module has three steps: document retrieval, candidate sentence extraction and initial score calculation. In this section we will describe the initial score calculation.

Suppose the candidate answer sentences set is $S = \{ s_1 , s_2 , \ldots, s_n \}$, the initial score of sentence s_i can be calculated by its target score and document score. The calculation of initial score is as follow,

$$initscore(s_i) = q * target score(s_i) + (1-q) * docscore(s_i)$$

Where we use *initscore*(s_i) for sentence s_i's initial score, *targetscore*(s_i) for sentence s_i's target score, *docscore*(s_i) for sentence s_i's document score and q for weight .

Target score is the corresponding score based on the words, phrases and entities, which occurred in the question target. It can be calculated as follow:

$$target score(s_i) = a \frac{c(w)}{n_w} + b \frac{c(p)}{n_p} + g \frac{c(e)}{n_e}$$

where n_w, n_p, n_e respectively represent the number of words, phrases and Name Entities contained in S_i, and $c(w)$, $c(p)$, $c(e)$ denotes the number of the words, phrases and Name Entities that in both S_i and the target. In our system, a, b and g are allotted to 0.3, 0.3 and 0.4 respectively.

The document score *docscore*(s_i) of the sentence s_i can be calculated as follow:

$$doc\,score(s_i) = Maxdocw(s_i) * (2 - \frac{2docn(s_i)}{docn(s_i)^2 + 1})$$

Where $docn(s_i)$ is the number of returned document including the sentence s_i, and $Maxdocw(s_i)$ is the max score of these documents. The document score increases apparently with $Maxdocw(s_i)$ and $docn(s_i)$.

Both target score and document score are normalized and set their weights empirically as 0.8 and 0.2 respectively, because the target information is more important than document information.

4.3 Relative Terms Extraction

Because getting web definition directly by question target is becoming a bottleneck, the system has to try other approaches to extract reliable information associated with target. Key phrase extraction and expansion are widely used in the text summarization. We utilized the technique for question target expansion. This process, called relative terms extraction, tried to obtain the words, phrases and entities that related with the target closely.

Given that $T=\{ t_1 , t_2 , \ldots, t_n \}$ are the all words, phrases and entities in candidate sentence set $S=\{ s_1 , s_2 , \ldots, s_n \}$, calculate the relativity $r(t_i)$ between t_i and target as follow formula:

$$r(t_i) = \sum_{j=1}^{n} E(t_i, s_j) * initscore\ (s_j)\ , \text{Where}\ \ E(t_i, s_j) = \begin{cases} 1 & t_i \in s_j \\ 0 & t_i \notin s_j \end{cases}$$

Rank T by $r(t_i)$, and select appropriate number of t_i as relative terms. In our system, we extracted top 15 relative words, phrases and entities which achieved the better performance.

Given that candidate sentence s_i has n_w words, n_p phrases and n_e entities. And these words, phrases and entities contained r_w relative words, r_p relative phrases and r_e relative related entities. The similarity between target and relative words, phrases and entities are denoted by $r(w_i)$, $r(p_i)$ and $r(e_i)$ respectively. Then the relative terms score $rwscore(s_i)$ of candidate sentence s_i can be calculated as below:

$$rwscore(s_i) = a * (\sum_{i=1}^{r_w} r(w_i)/n_w) + b * (\sum_{j=1}^{r_p} r(p_j)/n_p) + g * (\sum_{k=1}^{r_e} r(e_k)/n_e)$$

Because the entities are more important than general words and phrases, we assigned higher weight to entities. In our system, the weights, a, b and g are set as 0.3, 0.3 and 0.4 respectively by empirical.

4.4 Answer Ranking and Generation

Generally, candidate answers are excessive. Therefore these sentences could not be selected as answers directly. Synthetic method is used to rank and select these candidate sentences as question answers. In our system, every candidate answers are evaluated by the linear combination of three scores: initial score, web score [Zhang et al, 2005] and relative terms score.

In candidate answer ranking, we set these score weights dynamically with our evaluation system. After ranking these candidate sentences, redundancy removal will be done. And we choose the top 20 candidate sentences as the final question answers.

5. Results

We submitted three runs for the main task of TREC15 QA Track: FDUQA15A, FDUQA15B and FDUQA15C. In the three runs, the algorithms used to answer factoid questions are same; the only difference is we use different question series anaphora resolution results. The difference between the three runs of list questions described as follow: In FDUQA15C, we tried new method of answer type classification and new answer ranking metrics. In FDUQA15B, besides the new methods for answer type classification and answer ranking, we optimize the document searching strategies for list question answering. In FDUQA15A, we put all these new methods in FDUQA15C and FDUQA15B together and filter the redundant answers in the final answer list. Also we tune the parameters in each step to get its best performance in the test set of TREC 2005 93 list questions. As to definition questions, difference between the three runs is FDUQA15C use the last year's system, FDUQA15B is this year's system and FDUQA15A combines the results of FDUQA15C and FDUQA15B.

		FDUQA15A	FDUQA15B	FDUQA15C
Final Score		0.192	0.185	0.163
Factoid Question	#globally right	80	81	80
	#locally correct	3	4	3
	#unsupported	27	26	27
	#inexact	16	16	16
	#wrong	277	276	277
	Accuracy	0.199	0.201	0.199
List Question	Average F score	0.165	0.145	0.144
Definition Question	Average F score	0.223	0.222	0.159

Table 1 Performance of FDUQA Runs in TREC 2005

From Table 3, we can see that we get a lot of improvements of our list question answering system. That owes to the improvement of answer type classification, document searching, answer ranking and answer filtering. The algorithm we use to answer definition questions is quite promising.

There're still a lot of things to be improved in our list question answering system. The linear combination is too simple; some more sophistic methods can be used to improve the performance. The number of the answers for each question should have differed; however, we only simply pick out the equally top 15 answers for each question. The final score should

be more precise to divide the answers into two parts, one for the final answer, and the other for elimination. Some answer clustering methods also should be considered to improve the ranking performance.

Acknowledgements

This research was supported by the National Natural Science Foundation of China under Grant No. 60435020 and No. 60503070. We are very thankful to Zhongchao Fei, Feng Ji, Bo li, Yindong Chen and Chao Shen for their contributions in our work.

Reference

Lide Wu, Xuanjing Huang, Lan You, Zhushuo Zhang, Xin Li, Yaqian Zhou. FDUQA on TREC2004 QA Track. In: Proceedings of the Thirteenth Text REtreival Conference, Gaithersburg, Maryland, 2004

Lide Wu, Xuanjing Huang, Yaqian Zhou, Yongping Du, Lan You.2003. FDUQA on TREC2003 QA Task. Proceedings of the TREC-12, Gaithersburg, Maryland, 2003

Lide Wu, Xuanjing Huang, Junyu Niu, Yingju Xia, Zhe Feng, Yaqian Zhou.2002. FDU at TREC2002: Filtering, QA, Web and Video Tasks. Proceedings of the TREC-11, Gaithersburg, Maryland, 2002.

Lide Wu, Xuanjing Huang, Yaqian Zhou, Zhushuo Zhang, Feng Lin.2005. FDUQA on TREC 2005 QA Track. Proceedings of the TREC-14, Gaithersburg, Maryland, 2005.

Zhushuo Zhang, Yaqian Zhou, Xuanjing Huang, Lide Wu. Answering Definition Questions using web knowledge base. In: Proceedings of the 2nd International Joint Conference on Natural Language Processing, 2005

Hang Cui, Min-Yen Kan, Tat-Seng Chua, Jing Xiao. A comparative Study o Sentence Retrieval for Definitional Question Answering. In Proceedings of the 27th Annual International ACM SIGIR Conference, 2004

Vijay Krishnan, Sujatha Das, Soumen Chakrabarti. Enhanced Answer Type Inference from Questions using Sequential Models. In Proceedings of the Empirical Methods in Natural Language Processing 2005