Insun05QA on QA track of TREC2005

Yuming Zhao, ZhiMing Xu,Yi Guan, Peng Li

School of Computer Science and Technology, Harbin Institute of Technology,
Harbin, China,
Email :{ ymzhao, xuzm,guanyi, pli} @insun.hit.edu.cn

1 Introduction

This is the first time that our group takes part in the QA track. At TREC2005, the system we developed, Insun05QA, participated in the Main Task, which submitted answers to three types of questions: factoid questions, list questions and others questions. And we also submitted the document ranking which our answers are generated from.

A new sentence similarity calculating method is used in our Insun05QA system. It can be considered as an extension of vector space model. And our QA system incorporates several useful tools. These tools include WordNet, developed by Princeton University, Minipar by Dekang Lin, and GATE, developed by University of Sheffield. Moreover, external knowledge such as knowledge from Internet is also widely used in our system.

Since it is the first time that we take part in QA track and the preparing time is limited, we concentrate on the processing of factoid questions. And the methods we developed to process list and others questions are generated from the method used to process factoid questions.

The structure of our Insun05QA system will be describe in Section 2, the details of how we process the factoid, list and others questions will be introduced in Section 3, and our results in TREC2005 will be presented in Section 4.

2 Architecture of Insun05QA system

The architecture of Insun05QA system can be described by Figure.1. It is composed of 6 parts: Preprocess, including questions preprocess and documents preprocess, question analysis, document retrieval, sentence similarity calculation, Web retrieval and answer extraction.

In our QA system, GATE is used to help us accomplish name entity reorganization in preprocess of questions and documents. Minipar developed by Dekang Lin, is adopted to help making question analysis and it is also used to parse the relevant passage in answer extraction module. We also develop our document retrieval module based on SMART, an information retrieval system, developed by Cornell University in the 1960s.

In Insun05QA system, we obtain knowledge from the ontology in WordNet which help to

classify the answer type and the information from Internet that contribute to answer extractions. And we also use the synsets of WordNet to accomplish question extension and sentence similarity calculation.

The three kinds of questions, factoid, list and others, almost have the same processing flow, and differences are made inside the modules on the flow towards the three different kinds of questions.

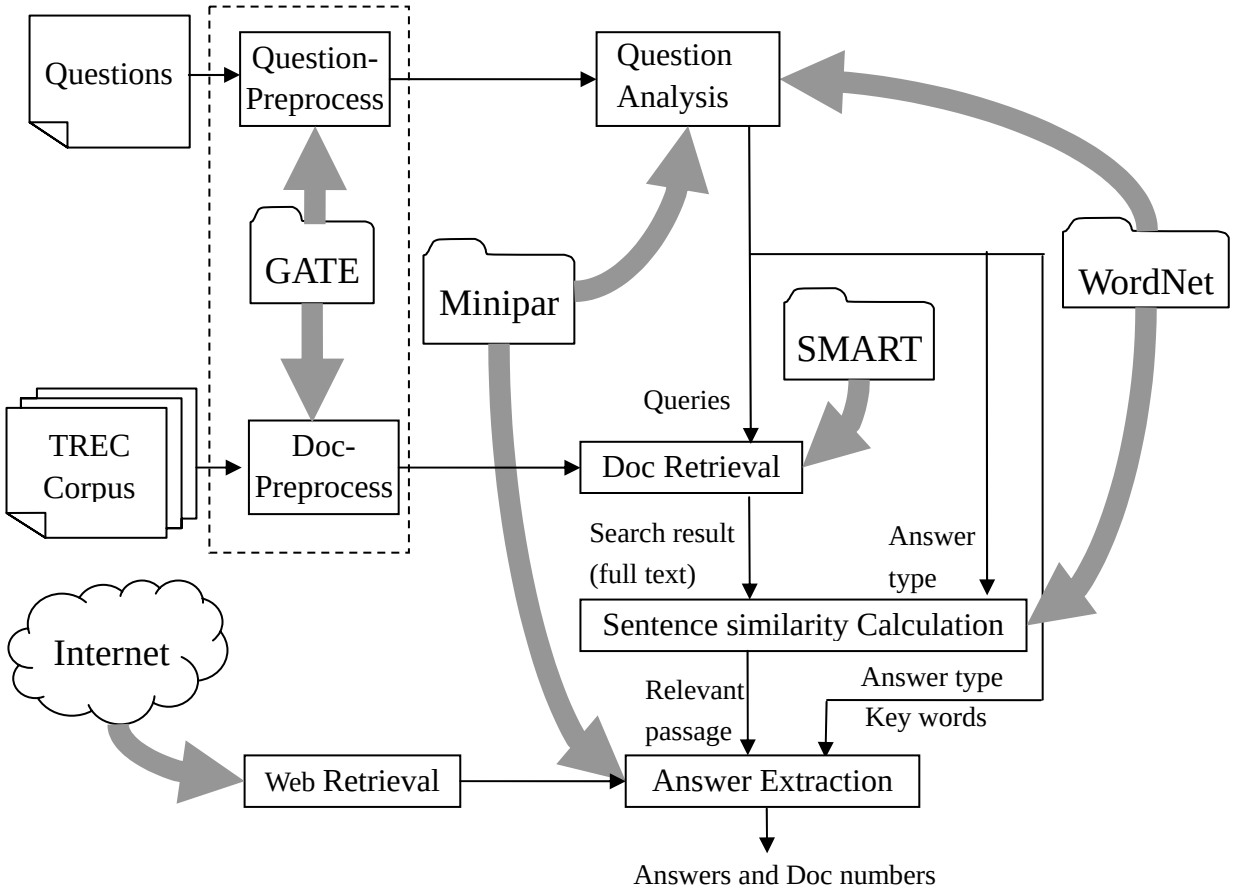

Figure.1. Flow Chart of Insun05QA

3 Main Components

3.1 Preprocessing

The preprocessing module is developed to accomplish the preprocess task for both documents and questions. The details of process steps are different for documents and questions. The flows of preprocessing for documents and questions are described separately by Figure.2 and Figure 3.

According to documents, the first process step is sentence boundary detect. In this step, we

detect the sentence boundary and write the text of documents in the form of one sentence a line. And after that, tokenization, part of speech, name entity recognization, stemming, and co-reference resolution, are applied orderly.

According to questions, there's no need of sentence boundary detect. The first thing we do is to classify the questions according to the type tag of each question in QA2005 test set, and we write the factoid, list and others questions into three files separately. And then, the following process steps, tokenization, part of speech, name entity recognization, and stemming, are applied, just like their being applied on documents. Things should be mentioned here is that when we rewrite the others questions into their own file, we generate the questions in a uniform pattern: **"What are the other matters about" + the target.**

The co-reference resolution for questions is different from documents. We accomplish co-reference resolution and rewrite the questions with the help of target information. For the question that target is about person, organization, or something like that, a simple algorithm is used to replace the pronoun and some short form with the question's target. But for the question that target is about events, a co-reference resolution algorithm that is relevantly complicated is applied. These methods make the questions holding enough information so that they will be instructive in the following steps.

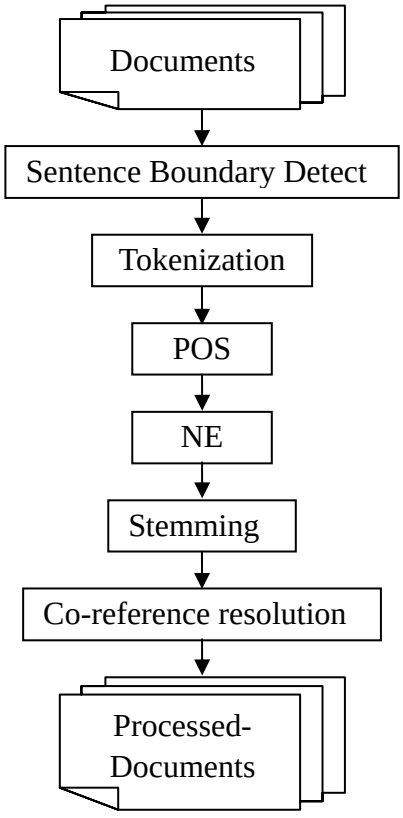

Figure.2. Flow Chart of Preprocess for documents

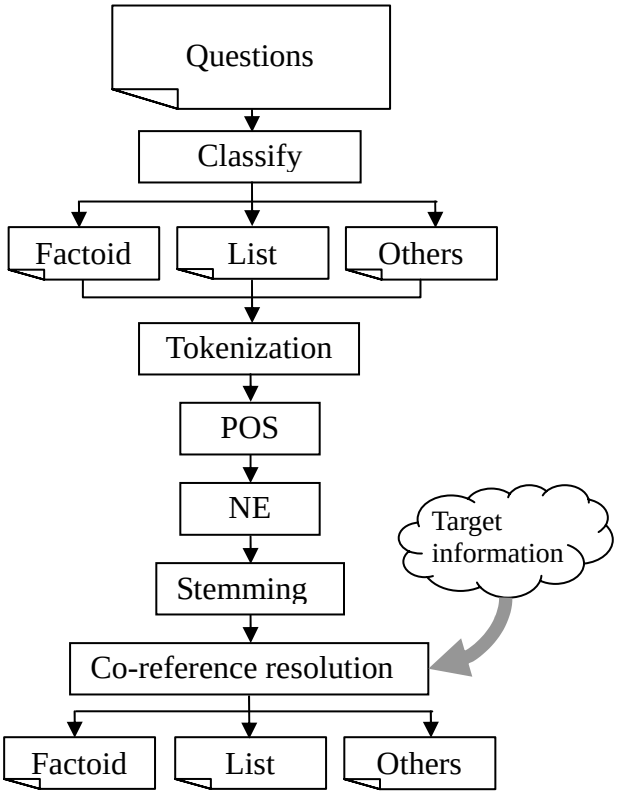

Figure.3. Flow Chart of Preprocess for questions

3.2 Question Analysis

The question analysis module has two functions: keyword generation and answer type prediction. We realize these functions based on two external tools: WordNet and Minipar.

We use Minipar to analyze the sentence structures of questions, and constituent information are extracted from questions based on this analysis. By using the synsets of WordNet, we build a dictionary of synonymous words, and with the help of this dictionary, we explore the constituents extracted from questions so as to accomplish keyword generation.

In the answer type prediction step, we adopt a rule-based algorithm to classify the answer type of input questions.

By studying the TREC questions of former years, we define a classification system which containing fifteen class answer types. These answer types are: person, location, organization, quality, date, ratio, money, ill, address, web address, job, measurement, color, film, and other.

Based on this fifteen-class answer type classification system, using questions of TREC 8-12 as training set, we get a library of rules by machine learning. These rules describe the relations between answer types of questions and their interrogatives and structures. In the process of learning these rules, constituents information of question sentences obtained in parsing above are used, and the ontology knowledge of WordNet is also used.

3.3 Documents Retrieval

In document retrieval module, relevant documents according to each question are retrieved from the TREC corpus. We build this module based on SMART, a retrieval system developed by Cornell University.

To carry out document retrieval, the index and inverse file of the TREC corpus to be dealt with are built at first. Since the file pointer of SMART is 32 bits, the size of the index and inverse file shouldn't exceed 2 GB. And then, for the same reason, documents that SMART could deal with in one time should not exceed 600MB or 800MB. The corpus of TREC2005 is about 3GB, so it's reasonable that we divide the TREC documents into 5 parts. We build index and inverse file for these 5 parts of documents separately, and relevant documents for every TREC2005 question are retrieved from these 5 parts of documents one by one. Then, we obtained the retrieval result by consolidating and re-ranking these 5 parts of retrieval results for every question.

3.4 Sentence Similarity Calculation

The sentence similarity calculation module is developed to extract the relevant passages of a question from the corresponding relevant documents obtained in the document retrieval module.

The core component of sentence similarity calculation module is the system similarity model (SSM), which can be considered as an extension of vector space model (VSM), and we use SSM to calculate the degree of similarity between the question and the sentence in relevant documents. The

system similarity theory is regarded as theoretical foundation of SSM. According to the system similarity theory, if there are some shared properties or features, two systems demonstrate some degree of similarity, and we call such two systems similar systems. These shared properties or features are called similar properties.

In SSM, Given a system $A=\{\alpha_1,\alpha_2,...,\alpha_m\}$, $m=|A|$ and a system $B=\{b_1,b_2,...,b_n\}$, $n=|B|$. Let $x_i>0$ denote the importance of component a_i $(1\le i\le m)$; Let $y_j>0$ denote the importance of component b_j $(1\le j\le n)$, assuming the number of the most similar binary tuple (MSBTs) is p $(p\le \min\{m,n\})$, denoted by $s_1, s_2, \ldots, s_p \in A\times B$, without loss of generality, supposing they are $<\alpha_1,b_1>,<\alpha_2,b_2>,...,<\alpha_p,b_p>$, with similarity degrees $\mu_i(1\le i\le p)$ respectively. The system similarity degree is $Q(A,B)$, then:

$$Q(A,B)=\frac{\sum_{i=1}^{p}\mu_i x_i^2}{\sqrt{\sum_{i=1}^{m}x_i^2}\sqrt{\sum_{i=1}^{p}\mu_i^2 x_i^2+\sum_{i=p+1}^{n}y_i^2}} \qquad (1)$$

In our sentence similarity calculation module, system denotes a sentence and elements such as $\alpha_1,\alpha_2,...,\alpha_m$ denote the words in a sentence. In Formula (1), x_i and y_j means the weights of according words in sentences. They are obtained by the method of variations of standard TF-IDF term weighting scheme. And the element similarity degree μ_i means the similarity degree between two similar words. We use information content to evaluate semantic similarity between words with the help of WorldNet, and then we obtained a list of words similarity degree which supports us carrying out Formula (1).

With SSM, we pick out sentences that are more similar to questions. The reason that we take out this step is that we consider that the answer always appears in or near the sentence that is much more similar to the question. But it is evident that the sentence containing answer isn't the most similar sentence to the question, for the answer won't appear in the question and there must be some un-similar elements in the answer sentence. And it is just these answer words that make the similarity degree between question and answer sentence decreased. It is contrary to what we expect, so we improve the SSM to make it adapt the requirement of our system.

We import a parameter λ to modify Formula (1), the improved $Q(A,B)$ is:

$$Q(A,B)=\frac{\sum_{i=1}^{p}\mu_i x_i^2+\lambda\sum_{i=1}^{q}y_i^2}{\sqrt{\sum_{i=1}^{m}x_i^2}\sqrt{\sum_{i=1}^{p}\mu_i^2 x_i^2+\lambda^2\sum_{i=p+1}^{p+q}y_i^2+\sum_{i=p+q+1}^{n}y_i^2}} \qquad (2)$$

With the information of answer type obtained in answer analysis module, we pick out the words of potential answer according to their name entity tags. We use λ $(\lambda\ge 0)$ to simulate the element similarity degree of these potential answer words, and thus make the words of potential answer contribute to the similarity degree between questions and answers. And using the former TREC QA data as training set, we get the proper λ by genetic algorithms.

We use this improved SSM calculating the similarity degree between questions and all the sentences in according retrieved documents. For each question, we rank the candidate sentences depending on their similarity degree. To avoid losing useful information, we expand each candidate sentence into a candidate passage which includes the sentence just before it and the sentence just after it.

3.5 Web Retrieval and Answer Extraction

Based on the results of former steps, the answer extraction module returns the exact answers and according documents numbers with the help of Web retrieval.

We begin our answer extraction with inspecting the candidate passages which is full of information including POS and NE tags. If the named entity in the candidate passages corresponds to an expected answer type, the entity will be picked out as a candidate answer. The selection of the best candidate answer, when multiple candidate answers exist, is performed with an answer ranking scheme that relies on heuristics method.

There is a wide, possibly infinite range of text features that can be designed to estimate the relevance of a candidate answer for the purpose of answer ranking. Besides using statistical features such as term frequency, proximity and relative position to the question key words, our methods also include syntactic information derived through parsing, and semantic features like word senses, POS tagging and keyword expansion etc. The following are some associated heuristic rules and relative features:

·Associated heuristic: if the candidate answer occurs in a passage with a large number of keywords or keywords expansion, then the candidate answer is more likely to be relevant.

F_{KN} = *Number* ({*the number of keywords / passage that candidate answer occurs}*)

F_{KEN} = *Number* ({*the number of keywords expansion / passage that candidate answer occur*})

·Associated heuristic: if the candidate answer occurs in the passage with a large number of different keywords or different keywords expansion, then the candidate answer is more likely to be relevant.

F_{KT} = *Number* ({*number of def-keywords/ passage that candidate answer occurs*})

F_{KET} =*Number* ({*number of def-keywords expansion/ passage that candidate answer occurs*})

·Associated heuristic: if the candidate answer is the subject or object of a sentence, then it is more likely to be relevant.

F_{SO} = (1, *if candidate answer is subject or object*)

·Associated heuristic: if the candidate answer is an apposition of a phrase containing many keywords, then it is more likely to be relevant.

F_{APP} = (1, *if candidate answer is apposition*)

·Associated heuristic: if the candidate answer is closer to the keywords or keywords expansion, then the answer is more likely to be relevant.

$$F_{DISK} = avg_{i=1}^{keywords} [distance\ (keyword_i, candidate\text{-}answer)]$$

$$F_{DISKe} = avg_{j=1}^{Exp-keywords} [\text{distance}\ (Exp - keyword_j, candidate\text{-}answer)]$$

·Associated heuristic: if the candidate answer appears several times in the candidate passage, it is more likely to be relevant.

F_{AT} = *Number* ({*times of candidate answer appeared in candidate passage*})

·Associated heuristic: if the candidate answer matches with answer extract pattern, it is more likely to be relevant. The answer extract patterns are built by studying the former TREC QA data.

F_{PAT} = (1, *if* answer extract pattern *is matched with*)

Our empirical answer ranking formula gives a score to every candidate answers by linear combination and weight adjusting. Eventually, a ranking of candidates according to the score is obtained.

To fetch up the information scarcity of documents, we develop the Web retrieval module to use the knowledge from Internet. Based on Google API, we developed a Web crawling tool, WebRept, which can download relevant web pages automatically. And based on some rules, the exact answer and document number are selected from the ranking of candidates, with the help of information returned by WebRept.

The answer extract method described above is used to deal with the factoid questions. For the list questions, the answer extract method is much similar. The difference is that for list questions, we return a set of answers selected from the ranking of candidates instead of only one answer. For the others questions, the answer extract method is completely different. Based on the ranking of candidate passages obtained in sentence similarity calculation, we get a new ranking of relevant documents. We use our InsunAbs system, an automatic document abstract generating system, on the relevant documents. After wiping off the information having appeared in the answers of the former questions which having the same target, the set of abstracts in the front of the ranking are returned as the answer of the other question.

4 Results

We have submitted only one run, Insun2005QA1, for the main task of TREC2005. The

evaluation result is described in Table 1.

Table 1. Performance of Insun2005QA in TREC2005

		Insun2005QA1
Average per-series score		0.187
Factoid questions	Number of correct	106
	Number of unsupported	15
	Number of inexact	16
	Number of wrong	225
	Accuracy	0.293
	Precision of NIL	0.057
	Recall of NIL	0.176
List questions	Average F	0.085
Others questions	Average F	0.079

From Table 1, we can see that the methods we use to deal with the list questions and others questions are too simple and arbitrary. Further research should be done to improve the performance of our system on dealing with the list and others questions.

Reference

[1] GUAN-Yi, WANG Xiao-long, WANG-Qiang. Measurement of System Similarity. JSCL-2005, Nanjing, Aug.2005

[2] G. Salton. Automatic Text Processing: The transformation, analysis, and retrieval of information by computer. Addison-Wesley, 1989

[3] Rohini Srihari and Wei Li. Question answering supported by information extraction. TREC-8 Proceedings, pages 75-85, 1999

[4] H. Cunningham. GATE, a General Architecture for Text Engineering. Computers and the Humanities, volume 36, pp. 223-254, 2002

[5] D. Lin. A Dependency-based Method for Evaluating Broad-Coverage Parsers. Proceedings of IJCAI-95. 1995

[6] Buckley, C., Singhal, A., and Mitra, M. Using Query Zoning and Correlation with SMART: TREC-5. In Proceeding of the 5th Text REtrieval Conference, NIST. 1996