FDUQA on TREC2003 QA task

Lide Wu, Xuanjing Huang, Yaqian Zhou, Yongping Du, Lan You
Fudan University, Shanghai, China

1 Introduction

It is the fourth time that we take part in the QA track. Our system, FDUQA, is based on our previous system (Wu et al, 2002). FDUQA includes an offline part and an online part. We make great efforts on the online part while leaving the offline part unchanged. We have tried many natural language processing techniques, and incorporated many sources of world knowledge, including Web. A novel Query formulation technique has also been put forward.

In addition, we've tried another attempt on answer extraction in this year's task. In the second section, we will describe the architecture of our QA system; and give a detailed description on the Query formulation for Web search in the third section; while in the fourth section, we will introduce our new attempt on answer extraction; and we will present our performance in the last section.

2 Architecture

FDUQA's architecture is shown in figure1. Our system can be divided in two ways. One is traditional: question analysis, retrieval, and answer extraction, as shown in figure 1 by the two horizontal lines. The other is more natural: answer type decision-making, candidate answer decision-making, and final answer decision-making, as shown in figure 1 by the two vertical lines. We'll describe the FDUQA system in the latter.

In the answer type decision-making step, FDUQA system determines the answer type of the input question based on the question's interrogative and focus words. The classifier and focus words decision algorithm are both based on heuristic rules. We adopt an eighteen-class answer type classify system, illustrated in table1. At this step our system can achieve the precision of 80%.

ABBR	NOUN_PHRASE	AGE
CAPITAL_WORDS	DESP_OF_ABBR	DATE
LOCATION	LENGTH	MEASURE
MONEY	NUMBER	ORGANIZATION
PERCENT	PREP_PHRASE	PERSON_NAME
SPEED	WRITING_NAME	NONTYPE

Table 1 Answer Type concepts

At the second step, candidate answer decision-making, our system searches the Web by Google and then tries to find the answer in the returned snippets. Finding an answer in the huge corpus is easier than in a smaller one in some sense, because system can search the corpus more easily and get more confident answer by stricter query (or query set). For example, questions such as "Where was Hans Christian Anderson born?" are very easy for Web search engine to find the answer by inputting query as "Hans Christian Anderson was born in". We'll describe the Query formulation for Web Search module in great details in the next section.

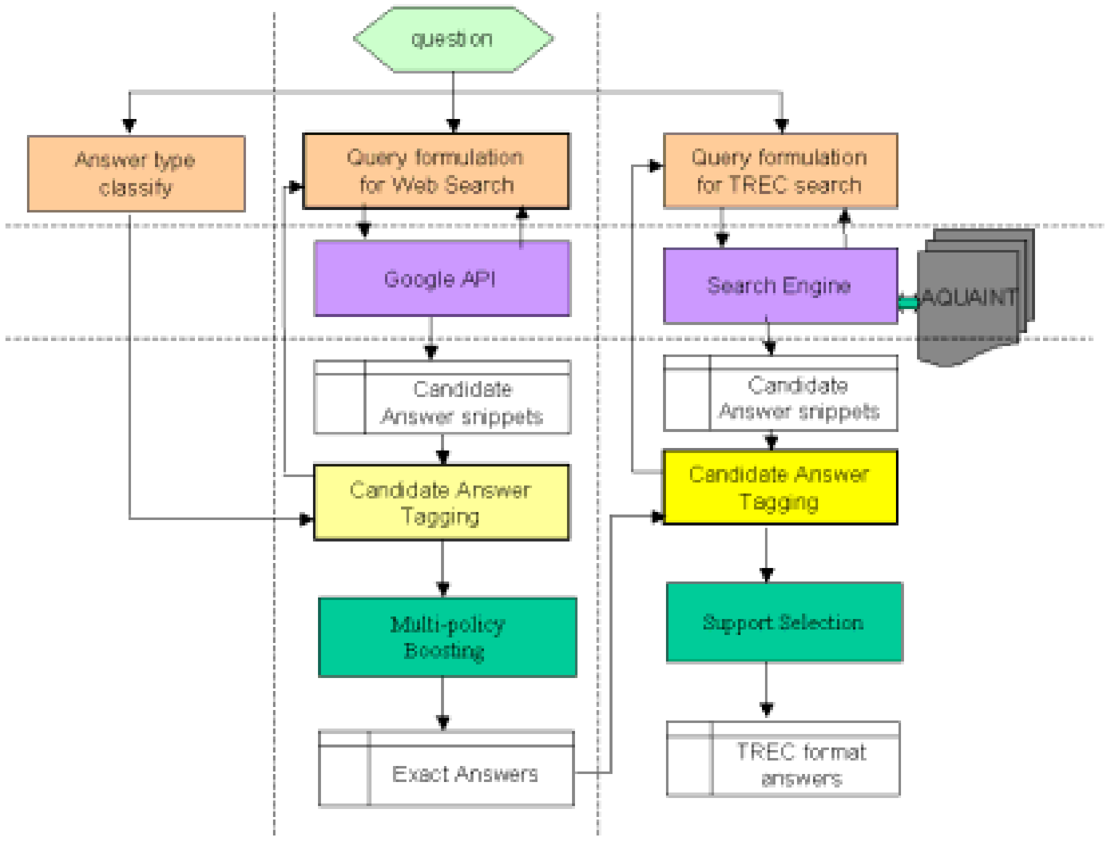

Figure 1 FDUQA system architecture

The upper two modules can be taken for question analysis and retrieval steps for the traditional QA system, while the following modules make up of the answer extraction step. Candidate answer tagging module tags candidate answers in the returned snippets based on their NE tagging and Base NP tagging results. Consider the distances of the key concepts and the distances between the candidate answers and key concepts, the multi-policy boosting module gives score to every candidate answer and snippet pair. The pairs are clustered by candidate answers, and each candidate answer set can get its score by adding up all of its elements' scores. Thus, the candidate answers can be sorted with their scores.

The third step is final answer decision-making section. At this step, the first two modules, Query generation for TREC search and search engine, are the same as last year. The following modules are almost the same as the corresponding modules in last step. The only difference between two "Candidate Answer tagging" modules is that system tagging the candidate answer in this step based on the candidate answer generated in last step, the candidate answer decision-making step. Support selection module sets the support score by adopting the same technique that used in multi-policy boosting module that give score to every candidate answer and snippet pair. The system integrates every candidate answer's support scores and their ranks in last step to sort them. FDUQA system considers the top one candidate answer as the final answer.

3 Query formulation for Web search

3.1 Query Formulation

Answer of a question may appear in a context, which is just the statement form of the question. For example, the answer of question “What book did Rachel Carson write in 1962?” appears in a context like “Rachel Carson wrote <Answer> in 1962”. But mostly such a context doesn’t exist in a limited corpus like AQUAINT. However, we do retrieval not only on the AQUAINT corpus, but also on Internet like some other systems (Kwok et al., 2001; Dumais et al., 2002). Because of the largeness and variety of information on Internet, this context can be retrieved now. Based on this idea, we formulate queries for Web retrieval. Figure 2 describes the process of query formulation.

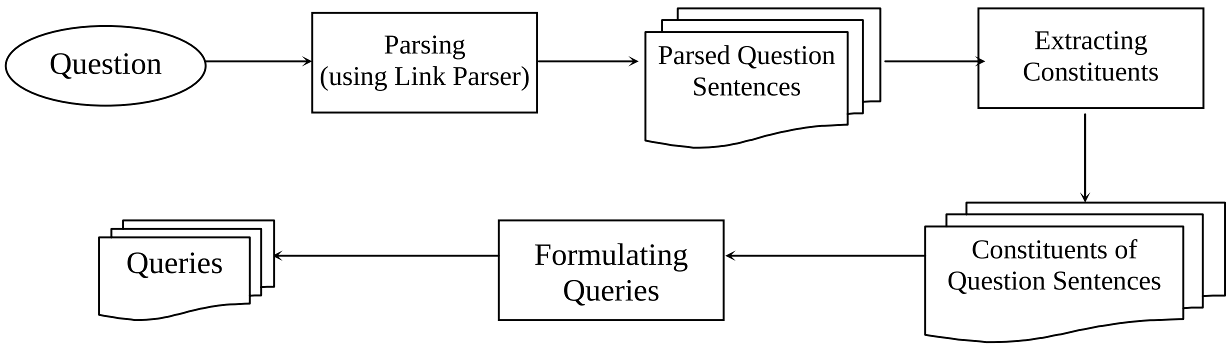
Question
Parsing (using Link Parser)
Parsed Question Sentences
Extracting Constituents
Constituents of Question Sentences
Formulating Queries
Queries

Figure 2 process of query formulation

Our system first parses questions using LinkParser (Sleator and Temperley, 1993), an English parser based on link grammar. Its precision is up to 0.9. Next we extract four constituents from the parsed question: subject, predicate, object and adverbial modifier. These constituents are then used to formulate queries for Web retrieval.

For example, we parsed the question “What book did Rachel Carson write in 1962?”

Its constituents are:

“Rachel Carson” – subject;
“wrote” – predicate;
“in 1962” -- adverbial modifier.

In this question, object of “wrote” is the question focus.

The queries formulated from the above constituents are:

"Rachel Carson wrote" "in 1962"
"Rachel Carson wrote" in 1962
"Rachel Carson" wrote in 1962
Rachel Carson wrote in 1962

Words in quote marks must appear continuously in retrieved snippets, while others may appear dispersedly or even not appear. Obviously, the first query is a tight one. And the followings are relatively looser. We generate loose queries allowing for other forms of context on Web. For example, “1962 Rachel Carson wrote Silent Spring that was aimed at the general public and became the Uncle Tom’s Cabin of the new environmentalism”. This snippet can’t be retrieved with the first query, but can be retrieved by the later three. And “...

Rachel Carson grew up on a small Pennsylvania farm, where she ... her degrees in 1932, she wrote science articles ... of the Sea, and finally Silent Spring in 1962" can only be retrieved by the last two queries.

3.2 Retrieval on Web

Among various Web search engines, we select Google because of its high performance. And the formulated queries are specified according to the requirement of Google. We submit queries to Google from tight ones to loose. Thus we can find snippets with answers for most of the questions.

We have done an experiment on questions of TREC2002 QA Track. In these 500 questions, TREC provided answers for 444 of them. So we only considered these 444 questions. And only the first 20 received results for each query are used. The result of Web retrieval is listed in table 2.

#question	#question (has answer in snippets)	#question (has answer in snippets and some snippet supports the answer)
444	367 (82.7%)	341 (76.8%)

Table 2 Web retrieval Result

We can find answers in the retrieved snippets for 82.7% of the 444 questions. And in a closer observation, the retrieved snippets support 341 answers, that is 76.8% of all these questions. Thus, most of the search results contain answers. It's important for the later processing.

4 New attempt for answer extraction

Pattern based method has been used by many other question answering systems, InsightSoft (Soubbotin and Soubbotin, 2001; Soubbotin and Soubbotin, 2001) has acquired good performance, ISI developed a method for learning patterns automatically (Ravichandran and Hovy, 2002).

We try a new pattern based method for implementing the answer extraction and give a solution to the problems that other system failed, such as only one key phrase of the question can be included within pattern. We will introduce the process of pattern learning and answer extraction with them.

The pattern for answer extraction is called context pattern, it is consisted of the following three parts: <Q_Tag>+[ConstString]+<A>. Here, <Q_Tag> stands for the key phrase in question, it includes different elements of the question, and we will introduce them later. <A> stands for the answer, any string holding the position will be extracted as the answer. "[ConstString]" is a sequence of words.

Context patterns can be learned automatically using the <Q_Tag , A> pairs as training examples. For instance, context pattern "<A>, Q_Focus of Q_NameEntity" can be used to answer the question "What is the capital of Syria?" "Q_Focus" represents the question term "the capital" and "Q_NameEntity" represents the question term "Syria".

We take the 500 questions of TREC 2002 as our training data for learning these context patterns.

4.1 Question Analysis

We define a set of notations to represent questions in advance as illustrated in table 3. They are the object or event the question asks about.

All these Q_Tag have different importance scores taking into account the possibility they appear around the answer.

The question pattern (Q_Pattern) is generated from its Q_Tag symbol set, and then the classification of questions will be built based on the Q_Pattern and the answer type. A case in point is that the question class “ [DAT] When was Q_BNP_1 Q_Verb? ” covered the question “ When was Apollo 11 launched? ”, “ When was the first atomic bomb dropped? ” and so on.

Q_Tag	Description	Importance_Score
Q_Quotation	the quotation part in the question	8
Q_Focus	the key word or phrase representing the object or event the question asks about (analyzed from Parser Minipar)	7
Q_NameEntity	the name entity in the question (analyzed from Name Entity tool)	6
Q_Verb	the main verb of the question (analyzed from Parser Minipar)	5
Q_BNP	the noun phrase of the question (analyzed from the BNP Chunking tool)	4

Table 3 Symbol Set of Question

4.2 Pattern Learning and Evaluation

We will explain our approach with the sample example below.

Sample question class: [LCN] What Q_Verb Q_Focus of Q_NameEntity?

Sample question: What is the capital city of New Zealand?

Where Q_Verb = “is”, Q_Focus = “the capital city”, Q_NameEntity = “New Zealand”, and Answer = “Wellington”.

The context patterns of each question class are learned by the following algorithm:

1. Constructing Query: “Q_Focus + Q_NameEntity +Answer” is constructed as the query. For example, the query of above sample question is: “the capital city”+“New Zealand”+ “Wellington”.
2. Searching: the query is submitted to the search engine Google and the top 100 Web documents are downloaded.
3. Snippet Selection and Filtering: the snippets for pattern learning are extracted from the Web documents. The answer, the nearest ten words left to it, and the nearest ten words right to it are retained.
4. Context Pattern Extraction: replace the question term in each snippet by the corresponding Q_Tag, and the answer term by the tag <A>. The minimum length string containing the Q_Tag and the tag <A> is extracted as the context pattern. For example, consider the string “…the number of languages that are being spoken. Wellington the capital city of New Zealand and …”, context pattern “<A> Q_Focus of Q_NameEntity” is extracted.

5. Computing the Initial Score of Context Pattern: the score is computed as the following formula considering the importance of the Q_Tag and the distance between the different Q_Tag and the answer. (α=1,β=0.6)

$$Initial\_Score = \alpha \bullet \frac{1}{Distance} + \beta \bullet \sum_{j=1}^{n} \frac{importance\_Score(QTag_j)}{importanceAll}$$

$$Distance = \frac{\sqrt{d_1^2 + d_2^2 + ... + d_n^2}}{n}$$

$$importanceAll = \sum_{k=1}^{m} importance(Q\_Tag_k)$$

m is the number of Q_Tag the question class contains, n is the number of the Q_Tag the context pattern contains, d$_i$ is the distance between the different Q_Tag and the answer.

The approach to context pattern evaluation is as follows. Query for each context pattern is formed and submitted to the Google, and the top 100 snippets are downloaded for context pattern precision calculation. The query consists of three parts: [Pre_Part]+[Post_Part]+[Q_Focus + Q_NameEntity].

[Pre_Part] stands for the word string left to tag <A> of the context pattern, and that [Post_Part] stands for the word string right to tag <A> of the context pattern. [Q_Tag] is composed of the Q_Focus and Q_NameEntity of the question. The matching score of each pattern is calculated as follows:

$$Match\_Score = \frac{Num_{Correct\_Match}}{Num_{Match}}$$

Num$_{Correct\_Match}$ denotes the number of snippets that tag <A> is matched by the correct answer; Num$_{Match}$ denotes the number of the snippets that tag <A> is matched by any word.

At last the score of the context pattern is computed with the formula : (α=0.3,β=0.7)

$$Pattern\_Score = \alpha \bullet Initial\_Scor + \beta \bullet Match\_Score$$

4.3 Answer Extraction

The context patterns can be used to extract answer to a new unseen question as follows:

1. Determine the question class of the unseen question based on its Q_Pattern and answer type. The corresponding context patterns are also selected.
2. Replace the Q_Tag symbols in the context pattern with the corresponding word string of the question.
3. For each context pattern and each snippet search engine returned, select the words matching tag <A> as the answer.
4. Sort the answers by their context pattern's score and their frequency.

The first answer is returned to the factoid question and the top five answers are returned to the list question and definition question.

5 Conclusion

This year we only take part in the main task of QA, and submit three runs. Our results are not very satisfactory. Our first run, FDUT12QA1, is based on our main architecture; FDUT12QA2 is our new attempt; and FDUT12QA3 is the simple combination of FDUT12QA1 and FDUT12QA2. Their detail evaluation report is illustrated in table 4.

	FDUT12QA1	FDUT12QA2	FDUT12QA3

Final score	0.163	0.122	0.165
Accuracy of factoid questions	0.194	0.179	0.191
Average F of list questions	0.088	0.067	0.086
Average F of definition questions	0.176	0.065	0.192
right questions of factoid questions	80	74	79
Unsupported questions of factoid questions	28	27	27

Table 4 Evaluation report

We find in table 4, that the numbers of unsupported questions of factoid questions are very big compared with their corresponding right answered questions. That's because we can't well integrate the Web into our system.

Acknowledgements

This research was partly supported by NSF of China under contracts of 69935010 and 60103014, as well as the 863 National High-tech Promotion Project of China under contracts of 2001AA114120 and 2002AA142090. We are thankful to Xin Li, Ningyu Chen, Liyuan Han for their help in the implementation.

References

Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web Question Answering: Is More Always Better? Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002), August 2002, Tampere, Finland.

Cody C. T. Kwok, Oren Etzioni and Daniel S. Weld. May 1-5, 2001. Scaling Question Answering to the Web. Tenth World Wide Web Conference. Hong Kong, China.

Deepak Ravichandran, Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. Proceedings of the ACL 2002.

Daniel Sleator and Davy Temperley. 1993. Parsing English with a Link Grammar. Third International Workshop on Parsing Technologies.

Soubbotin, M.M. AND Soubbotin. 2001. Patterns of Potential Answer Expressions as Clues to the Right Answer. Proceedings of the TREC-10, Gaithersburg, Maryland, 175-182.

Martin M.Soubbotin, Sergei M.Soubbotin. 2002. Use of Patterns for Detection of Likely Answer String: A Systematic Approach. Proceedings of the TREC-11, Gaithersburg, Maryland, 134-143.

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.