Web based Pattern Mining and Matching Approach to Question Answering

Dell Zhang[1,2]
[1] Department of Computer Science
School of Computing
S15-05-24, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2] Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744251
dell.z@ieee.org

Wee Sun Lee[1,2]
[1] Department of Computer Science
School of Computing
S15-05-24, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2] Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744526
leews@comp.nus.edu.sg

Abstract

We describe herein a Web based pattern mining and matching approach to question answering. For each type of questions, a lot of textual patterns can be learned from the Web automatically, using the TREC QA track data as training examples. These textual patterns are assessed by the concepts of support and confidence, which are borrowed from the data mining community. Given a new unseen question, these textual patterns can be utilized to extract and rank the plausible answers on the Web. The performance of this approach has been evaluated also by the TREC QA track data.

1 Introduction

What a current information retrieval system or search engine can do is just "document retrieval", i.e., given some keywords it only returns the relevant documents that contain the keywords. However, what a user really wants is often a precise answer to a question. TREC has launched a QA track to support the competitive research on question answering, from 1999 (TREC8). The focus of TREC QA track is to build a fully automatic open-domain question answering system, which can answer factual questions based on very large document. Today, the TREC QA track [V0099][Voo00][Voo01] is the major large-scale evaluation environment for open-domain question answering systems.

Most of the recent question answering systems [V0099][Voo00][Voo01] require sophisticated linguistic knowledge or tools, such as parser, named-entity recognizer, ontology, WordNet, etc. However, at the TREC10 QA track [Voo01], the best performing system just used many textual patterns [SS01]. The power of textual patterns for question answering looks quite amazing and stimulating to us.

We describe herein a Web based pattern mining and matching approach to question answering. For each type of questions, a lot of textual patterns can be learned from the Web automatically, using the TREC QA track data as training examples. These textual patterns are assessed by the concepts of support and confidence, which are borrowed from the data mining community. Given a new unseen question, these textual patterns can be utilized to extract and rank the plausible answers on the Web. The performance of this approach has been evaluated also by the TREC QA track data.

To illustrate our approach, we would like to use the question "Who was the first American in space?" as a running sample in the following sections. This question was the No.21 test question in the TREC8 QA track.

2 System Overview

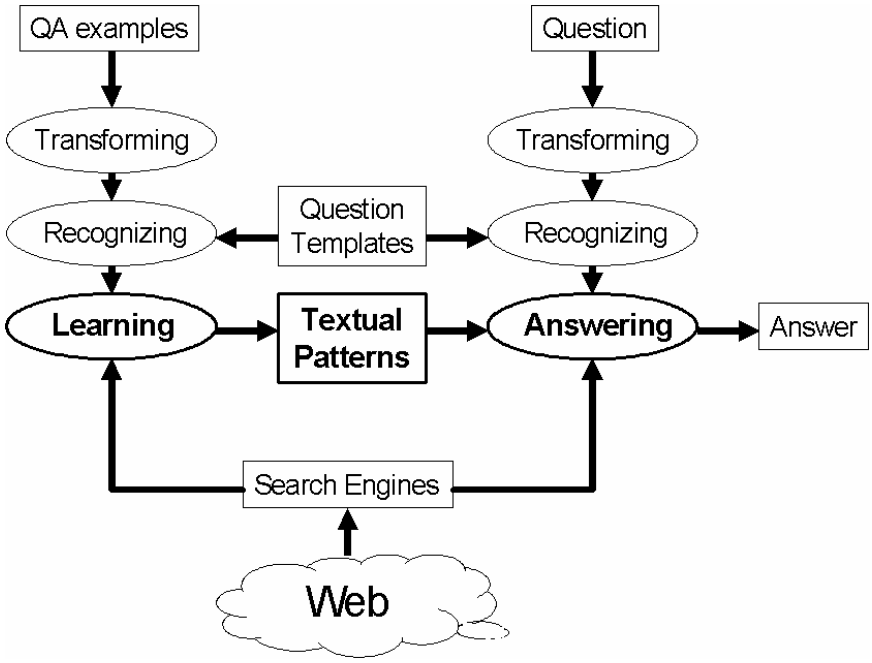

Figure 1, system architecture.

As shown in Figure 1, the system entails two main functions, one is learning and the other is answering. For both functions, the question has to be pre-processed by the transforming and recognizing module.

The answering part of the system relies on the textual patterns. A textual pattern can be in either of the following two forms.

_Q_ <intermediate string> _A_ <boundary string>
<boundary string> _A_ <intermediate string> _Q_

Here, **_Q_** stands for the question key phrase and **_A_** stands for the potential answer. The key phrase of a question is a continuous sequence of words in that question, which represents the primary object or event the question asking about. For instance, the key phrase of the sample question would be "the first American in space". A textual pattern actually describes the context of some potential answers to a specific class of questions.

Such textual patterns can be learned using some question-answer pairs as training examples. For instance, the correct answer to the sample question can be found in the string "In 1961, Alan Shepard became the first American in space ……", this observation suggests that the textual pattern "**, _A_ became _Q_**" can be used to answer similar questions like "Who was the first U.S. president?", "Who was the second man to walk on the moon?", and so on.

The learning part of the system will take advantage of the TREC QA track data as training examples. In each year's competition, TREC organizers issue several hundred questions to test the participant systems, and later release the regular expressions indicating the correct answers to each test question. Such TREC questions along with their answer regular expressions are our training examples.

3 Algorithms

3.1 Transforming

The transforming algorithm attempts to guess how the answer to the question may appear in a target sentence, i.e. a sentence that contains the answer, and then transforms the question to that format. We hope there will be more chances to find the answer after transforming the question. Currently two transforming methods are used.

For questions with an auxiliary do-verb and a main verb, the target sentence is likely to contain the verb in the conjugated form rather than separate verbs. For instance, the answer to the question "When did Nixon visit China?" would more likely to occur in the target sentence as "…… Nixon visited China ……" rather than "…… did Nixon visit China ……". So we transform the question by conjugating the auxiliary do-verb and the main verb. To locate the main verb like "visit" in the question, we have to parse the question using the MEI parser [Cha99]. To find the converted verb like "visited", we utilize PC-KIMMO [Ant90].

For questions with an auxiliary be-verb and a main verb (past particular), the target sentence is likely to contain these two verbs continuously together. For instance, the answer to the question "When was the telephone invented?" would more likely to occur in the target sentence as "……the telephone was invented ……" rather than "……was the telephone invented ……". So we transform the question by moving the auxiliary be-verb to the place just before the main verb. Again the MEI-parser is used to locate the main verb.

3.2 Recognizing

The recognizing algorithm determines the class of the question and the key phrase of the question. This was done by finding the appropriate template of the question.

Currently, we have defined 22 question classes, and each class has several templates formulated as regular expressions which indicate the possible appearance of this type questions. For instance, some of the templates in the ACRONYM class are as the following.

What is _Q_
What is the meaning of _Q_
What the (?:acronym|abbreviation) _Q_ (?:stands for|means)
What _Q_ (?:stands for|means)
What the initials _Q_ (?:stand for|mean)
_Q_ is (?:an|the) (?:acronym|abbreviation) for what
_Q_ stands for what

3.3 Learning

Assume the set of QA examples is E. Each QA example $e_i = (q_i, rea_i)$ consists of two parts, the question q_i and the regular expression of its correct answer rea_i. The following algorithms can learn the textual patterns of a specific question class c from the Web.

3.3.1 Construct Snippet Database

Given the QA examples of class c, $E^{(c)} = \{(q_i, rea_i) \mid (q_i, rea_i) \in E \wedge class(q_i) = c\}$, the following algorithm can automatically construct the snippet database of class c, $S^{(c)}$.

For each $(q_i, rea_i) \in E^{(c)}$**, do {**

Submit $kp(q_i)$ **as a phrase along with the words in** q_i **to a Web search engine, such as Google (http://www.google.com/).**
Grab the search result returned by the search engine.

Extract the text snippets from the search result.
For each snippet s_{ij}, do {
Insert a special symbol "_<_" at the head of s_{ij}.
Append a special symbol "_>_" at the tail of s_{ij}.
Replace every $kp(q_i)$ with a special symbol "_Q_".
Find the answer a_i using the answer regular expression rea_i.
Replace every a_i with a special symbol "_A_".
Add s_{ij} to $S^{(c)}$.
}
}

For instance, the sample question "Who was the first American in space?" and its answer regular expression "((Alan (B\.)?)?Shephard)" can be taken as a QA example of the WHO-IS class. The following query will be submitted to Google, and then get the search result.

"the first American in space" Who was the first American in space?

A small portation of the snippet database of the WHO-IS class is shown in Figure 2.

.....
_<_ in mind that Alan Shepard was _Q_, not the first man. _>_
_<_ John Glenn is not picked to be _Q_. _>_
_<_ Alan Shepard becomes _Q_. _>_
_<_ Then on 5 May 1961, less than a month after the Gagarin mission, Alan Shepard became _Q_. _>_
_<_ with general relativity theory to follow, in 1905 60 Yuri Gagarin became the first man in space in 1961 66 Alan Shepard became _Q_ in Yahoo! _>_
_<_ _Q_ was Alan Shepard, launched on May 5th Sea and Sky: Space Exploration 1961 - 1970. _>_
.....

Figure 2, sample snippet database

Two special symbols "_<_" and "_>_" are used to simplify the algorithm, where "_<_" stands for the head of the snippet and "_>_" stands for the tail of the snippet.

3.3.2 Discover Textual Patterns

Given the snippet database of class c, $S^{(c)}$, the following algorithm can automatically discover the textual patterns of class c, $P^{(c)}$.

For each $s_{ij} \in S^{(c)}$, do {
If (s_{ij} contains both "_Q_" and "_A_") then {
Extract the textual pattern p_k from the snippet using the following two regular expressions:
(\b_Q_\b(.*?)\b_A_\b\s*(_>_|\W|(\w+)))
((_<_|\W|(\w+))\s*\b_A_\b(.*?)\b_Q_\b)
If (p_k is not in $P^{(c)}$ yet) then Add p_k to $P^{(c)}$.
}
}

Some of the discovered textual patterns of the WHO-IS class is shown in Figure 3.

```
.....
, _A_ became _Q_
_<_ _A_ was _Q_
_Q_ was _A_,
_A_ made history as _Q_
by _A_ ( _Q_
_Q_ , _A_ _>_
.....
```

Figure 3, sample discovered textual patterns

3.3.3 Assess Textual Patterns

Given the textual patterns of class c, $P^{(c)}$, and the snippet database of class c, $S^{(c)}$, the following algorithm can automatically assess the textual patterns in $P^{(c)}$.

For each textual pattern $p_k \in P^{(c)}$, do {

Translate the textual pattern p_k into a regular expression $re(p_k)$, the special symbol "_A_" are replaced by "(.*?)", means this part can be matched by any string.

Search $re(p_k)$ in $S^{(c)}$.

Let X denote the set of snippets which can match $re(p_k)$.

Let Y denote the set of snippets which can not only match $re(p_k)$, but also the string corresponding to the "(.*?)" part is just "_A_ ".

$$\text{support}(p_k) = \frac{|Y|}{|S^{(c)}|},\ \text{confidence}(p_k) = \frac{|Y|}{|X|}$$

If $\text{support}(p_k)$ is less than the threshold t_{support}, then remove p_k from $P^{(c)}$.

If $\text{confidence}(p_k)$ is less than the threshold $t_{\text{confidence}}$, then remove p_k from $P^{(c)}$.

}

In fact, a textual pattern can be considered as an association rule "context => answer". The concepts support and confidence are borrowed from the data mining community.

Some of the assessed textual patterns of the WHO-IS class is shown in Figure 4.

.....	*confidence*
, _A_ became _Q_	0.09
_<_ _A_ was _Q_	0.11
_Q_ was _A_,	0.05
_<_ _A_ made history as _Q_	1.00
by _A_ (_Q_	0.66
_Q_ , _A_ _>_	0.14
.....	

Figure 4, sample assessed textual patterns

3.4 Answering

Having the assessed textual patterns of each question, the following algorithm can be employed to answer a new unseen question.

For a new question q_{new}, determine its class c and its key phrase $kp(q_{new})$, by transforming and recognizing algorithms.

Submit $kp(q_{new})$ as a phrase along with the words in q_{new} to a Web search engine, such as Google (http://www.google.com/).
Grab the search result returned by the search engine.
Extract the text snippets from the search result.
For each snippet s_j, do {

Insert a special symbol "_<_" at the head of s_j.

Append a special symbol "_>_" at the tail of s_j.

Replace every $kp(q_{new})$ with a special symbol "_Q_".

For each textual pattern $p_k \in P^{(c)}$, do {

Translate the textual pattern p_k into a regular expression $re(p_k)$, the special symbol "_A_" are replaced by "(.*?)", means this part can be matched by any string.

If s_j can match $re(p_k)$, then {

Take the string corresponding to the "(.*?)" part as a plausible answer a_{jk}.

If (a_{jk} is not in $A^{(q)}$ yet) then {

Add a_{jk} to $A^{(q)}$

Let $\text{confidence}(a_{jk}) = \text{confidence}(p_k)$

}

else {

Increase $\text{confidence}(a_{jk})$ **by** $\text{confidence}(p_k)$

}

}

}

}

Remove the unreasonable answers in $A^{(q)}$ using the stop-answers list and class-specific filters.
Sort all the found answers in $A^{(q)}$ by their confidence value.
Return the ordered top-*N* answers in $A^{(q)}$.

For instance, the answer to the sample question can be extracted from the snippet "Alan Shepard was the first American in space ……" with confidence 0.11 using a textual pattern of the WHO-IS class "**_<_ _A_ was _Q_**".

The list of stop-answers contains the strings which have no chance to be a correct answer in our opinion, such as "he", "today", "http", etc. And for each class of questions, we apply a class-specific filter which can help to remove the answers not for this class, e.g., a date class filter rejects a location string even it matches the textual pattern.

4 Experiments

Several experiments have been done to evaluate the performance of this approach, using the data from TREC8, TREC9 and TREC10. The questions with typo mistakes, the definition style questions like "Who is Colin Powell?", the questions which are syntactic rewrites of earlier questions (TREC9 test questions No.701-893), and the questions with no associated answer regular expressions have been removed from the data set. Note all the Web search results were retrieved from Google in the period July -- August 2002.

Test Data	*t#*	*Train Data*	*e#*	*r#*	*c#*	*MRR_all*	*MRR_ret*
TREC8	196	TREC9,10	757	93	52	0.22	0.46
TREC9	438	TREC10,8	515	282	155	0.27	0.42
TREC10	319	TREC8,9	634	230	120	0.28	0.39
TREC8,9,10	953	TREC8,9,10	953	634	509	0.53	0.79

Table 1, the performance of this approach on TREC8, TREC9 and TREC10 questions.

The experiment results are shown in Table 1. Here *t#* means the number of test questions, *e#* means the number of training examples, *r#* represents the how many questions the system has returned some answers for, *c#* represents how many questions the system has correctly answered. The MRR (Mean Reciprocal Rank) metric was used in TREC8, TREC9 and TREC10. Here *MRR_all* represents the MRR score over all test questions, while *MRR_ret* represents the MRR score over the questions which the system has returned some answers for.

The MRR score of this approach is not as high as that of the best question answering system in TREC. This discrepancy is due to many reasons. One important factor is that the answer regular expressions provided by TREC are quite limited, many correct answers such as "Alan B. Shepard, Jr." are judged wrong since they do not occur in the TREC specified document collection. Another interesting issue is time, the correct answers to some questions like "Who is the U.S. president?" will change over time. The Web is also messier than the TREC document collection.

This approach works quite well on simple questions, e.g. questions about ACRONYM, AUTHOR, BIRTHDATE, etc. The overall MRR score (MRR-all) for AUTHOR questions in TREC8 are above 0.55, while using the AUTHOR questions and answers in TREC9 and TREC10 as training examples. The performance of LAMP will dramatically drop down when the length of the question becomes longer, for instance, a TREC8 question averagely contains 9.93 words, but a correctly answered TREC8 question averagely contains 6.75 words.

The performance of this approach on TREC11 questions is as follows.

CWS (Confidence Weighted Score): 0.458
Precision of recognizing no answer is 39 / 293 = 0.133
Recall of recognizing no answer is 39 / 46 = 0.848

The above experiment results imply that this approach has high precision but low recall, i.e., this approach prefers "no answer" rather than "wrong answer". We think this property is good because "wrong answer" is usually worse than "no answer". And, this property allows this approach to be easily augmented with other approaches.

To increase recall for our TREC 11 entry, we augmented this approach with a simple answer extractor based on Support Vector Machines (SVM) [CS00] trained using features constructed from words in the question, words in the candidate sentence and the POS (Part-of-Speech) tags of words neighboring a candidate exact answer. To find a supporting document, we return the highest ranked document (in the list returned by the organizers) that contains the answer returned by the system.

The performance of the hybrid system on TREC 11 questions is as follows.

CWS (Confidence Weighted Score): 0.396
Precision of recognizing no answer is 0 / 2 = 0.000
Recall of recognizing no answer is 0 / 46 = 0.000

5 Conclusion

This approach distinguishes itself by its simplicity. It just uses the snippets in the Web search results, since it is time-consuming to download and analyze the original web documents. It does not require any sophisticated natural language processing on snippets, and does not need any advanced data structure, just simple hash table is enough. Because of this simplicity, LAMP is very efficient, which makes it perfect for online question answering.

One limitation of this approach is that one textual pattern can include only one question key phrase in the current stage. It does not work for the questions having multiple key phrases, possibly apart from each other. For example, to answer the question "How many calories are there in a Big Mac?", it would be better to use two question key phrases, "calories" and "Big Mac". Another drawback is that the textual patterns cannot handle long-distance relationships between the question key phrase and the answer. For example, the textual pattern "**_Q_ became _A_**," cannot locate the answer in the text snippet "Alan Shepard, who in 1961 became the first American in space and ……". However, the abundance and variation of the Web information makes it feasible to find answers on the Web with high probability through this approach, because the factual knowledge is usually replicated across the Web, expressed in many different forms [BLB+01].

References

[Ant90] E. L. Antworth. *PC-KIMMO: a two-level processor for morphological analysis.* Dallas, TX: Summer Institute of Linguistics, 1990.

[Cha99] E. Charniak. *A Maximum-Entropy-Inspired Parser*. Technical Report CS-99-12, Brown University, Computer Science Department, August 1999.

[CS00] C. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, Cambridge, UK, 2000.

[BLB+01] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-Intensive Question Answering. In In *Proceedings of the 10th Text Retrieval Conference (TREC10)*,, pp.183-189, NIST, Gaithersburg, MD, 183- 189, 2001.

[SS01] M. M. Soubbotin and S. M. Soubbotin. Patterns of Potential Answer Expressions as Clues to the Right Answer. In *Proceedings of the 10th Text Retrieval Conference (TREC10)*, pp. 175- 182, NIST, Gaithersburg, MD, 2001.

[Voo99] E. Voorhees. The TREC-8 Question Answering Track Report. In *Proceedings of the 8th Text Retrieval Conference (TREC8)*, pp. 77-82, NIST, Gaithersburg, MD, 1999.

[Voo00] E. Voorhees. Overview of the TREC-9 Question Answering Track. In *Proceedings of the 9th Text Retrieval Conference (TREC9)*, pp. 71-80, NIST, Gaithersburg, MD, 2000.

[Voo01] E. Voorhees. Overview of the TREC 2001 Question Answering Track. In *Proceedings of the 10th Text Retrieval Conference (TREC10)*, pp. 157-165, NIST, Gaithersburg, MD, 2001.