Abstract

The basic architecture of a Question Answering System (QAs), based on Natural Language Processing, subsumes question analysis and answer extraction. The paper presents a system which is based on semantic analysis, relates the words logically and provides an admissible answer to the user query. Instead of using template based query, it accepts questions phrased in various forms. The question is analyzed semantically to reduce it to a canonical form expressed as a dependency tree. It extracts the answer by analyzing the question depending upon the rules formed from the dependency tree and searching through the generated state graph using certain heuristic. Results of the evaluation, done on TREC-10 test set, shows a significant enhancement in the efficiency of the system.

1. Introduction

It is a common experience that complex queries are difficult to formulate using information retrieval systems that use only keywords for framing the query. A more satisfactory approach would be to allow the user to frame the query in the form of a question written in natural language. In such Question Answering systems (QAs), a user poses the query in natural language and the system finds the most precise and concise answer for it from the given corpus. The QAs can be for Open or Closed Domain Systems. This paper deals with open domain QAs which have lower accuracy because the vocabulary is unlimited and a particular word usually comes in more than one sense. The evaluation of the result obtained is also equally challenging because of the same reasons [1].

Extensive research has already been done in this field which comprises of keyword matching (Dongfeng Cai et al) [2], rule based matching (Riloff et al) [3], semantic web (Andreas et al) [4], ontology (Jibin Fu et al) [5], semantic reformulation (M Ramprashath et al) [6], template based (D.S Wang et al) [7]. The previous research done on this problem has resulted in systems like FREyA (Damljanovic et al) [8], ORAKEL (Cimiano et al) [9], SWSE (Andreas Harth et al) [4].

SWSE (Andreas Harth et al) [4] though is a web based system it has not worked on question analysis. Thus questions phrased in complex form do not return the correct answer. Also there is RDF generation but searching is only based on keywords. FREyA (Damljanovic et al) [8] a Feedback Refinement and Extended Vocabulary Aggregation system uses syntactic parsing. In FREyA, ontology is applied on user query with the help of the user’s feedback. Although the Precision and Recall value for the tested data reaches a high of 92.4% it asks user each time for suggestion. Also it missed certain entries while matching based on fuzzy logic. ORAKEL (Cimiano et al) [7] based on knowledge represented in natural language mapped on ontology structures, handles only wh-based questions. The lexicon needs to be updated constantly for efficient mapping. The system fails straight away for a grammatically incorrect input. There are systems based on semantic reformulation (D.S Wang et al) [7] which formulate the user query with the help of the argument extracted from the query and answer pair and retrieves the answer from the relevant documents extracted using web search engine. But such systems cannot answer non-factoid type of questions.

In contrast to the earlier approaches, the current system is able to analyze the question in order to surmise the actual need of the user. This is difficult since the same question may be framed in many different ways. The question analyzer of this module, unlike other previously developed systems, works on the question in a semantic way, no matter how complex its phrasing is. Instead of simply using a template based approach for question analysis, it understands the relation between the words i.e. their dependencies [10] and returns the expected answer type.

At the second level, the answer searching system has to sift through the corpus and retrieve the correct sections of the relevant documents using paragraph ranking based on weighted keywords. The answer retrieval system further generates parse tree of the relevant documents in the corpus using Stanford parser [11]. It then generates a state graph from the parse tree, which also include cause-effect relationships and can potentially span an entire document. Thus, sentences that are physically far apart but related directly or indirectly by some common entity will acquire direct edges in the graph. For returning the answer it uses a heuristic approach and searches according to the expected answer type, as per the question. This approach efficiently deals with both factoid as well as non-factoid type of
questions and is immune to common trifling grammatical errors.

The next section contains the methodology. The two basic modules of the system i.e. the question analysis and answer extraction are explained in that section. Semantic analysis of questions is done and rules are formulated from the dependency tree \([10]\). The question Analyzer thus categorizes the type of answer expected and for Answer Retrieval, paragraphs are ranked using biased entity-verb matching. A State Graph, including cause effect relationships, is generated and the answer is searched using a heuristic based approach. The later sections contains the result of the evaluation done on TREC-10 test set as well the analysis of the results.

2. Proposed Methodology

Our proposed methodology relies on three basic modules: corpus preparation, question analysis and answer extraction. Firstly, the lexical database is generated which helped in classification of entities during question analysis and corpus analysis. And entity database is prepared from corpus database using Stanford NER which helps later in answer extraction.

In the question analysis phase, different algorithms are designed to deal with all type of question using dependency tree generated from the Stanford Parser which anticipated the expected answer type. Once we received the expected answer type from question analysis, paragraphs are extracted from the corpus which are further ranked depending upon the user question. We have defined the rules to generate the state graph which helps in answer extraction using certain heuristic based approach. The details of each module is described in the following subsections.

2.1 Preparing Corpus and Lexical Database

The corpus data i.e. set of files, is tagged using the Stanford NER \([13]\) and the answer analyzer uses it to search for the answer. Also, lexical database is fabricated which is referred for classification of entities during question and corpus analysis (in graph generation to assign descriptor to the various nodes).

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animal</td>
<td>Cobra, Phoenix, Crocodile</td>
</tr>
<tr>
<td>Location</td>
<td>Europe, Switzerland, Germany</td>
</tr>
<tr>
<td>Time</td>
<td>Morning, June, A.M., Sunday</td>
</tr>
<tr>
<td>Number</td>
<td>Rupees, Liter, Mile</td>
</tr>
<tr>
<td>Organization</td>
<td>Microsoft, ACM, UNO</td>
</tr>
<tr>
<td>Language</td>
<td>Hindi, English, Spanish</td>
</tr>
</tbody>
</table>

Table 1: Representing Descriptor Entity Relationship.

2.2 Question Analysis

The question analysis part anticipated the expected answer type from the question and is implemented as two modules: first module parsed the query and formulated a dependency tree out it and second part retrieved the expected answer type from the dependency tree.

2.2.1 Query Parsing

Query is parsed to generate the dependency table using Stanford Parser which depicts the graphical relationship between the words in the question.

2.2.2 Classification and expected answer type

The dependency table is processed to return the classifier word using the rules which deal with all type of questions as shown in Fig. 2. The hyponym (from Word Net) of the classifier word is used to get the expected answer type and classified it into person, location, time, number, organization and miscellaneous entities using lexical database. The detailed description of the algorithm is described in the Fig. 2.
Figure 2: Question Processing: Identify the tag type corresponding to the question word present in user query and then find “pairword” using tag type and question word from the dependency table generated using Stanford Parser. Finally this algorithm will give the “classifier” word which will mark the category of the expected answer from the query. E.g. time, place, reason etc.

As has been mentioned previously, in the present work we do not restrict the user to formulate the query in a particular manner. Hence, the same question can be phrased in different ways. Thus, the purpose of this module is to return the expected answer type, irrespective of the phrasing of the query.

An example is presented below which shows how we get location as the expected answer type from the questions phrased in various forms.

**Question 1: What is the name of the place where Gandhi was born?**

<table>
<thead>
<tr>
<th>Tag Type</th>
<th>Dependency Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attr</td>
<td>is, what</td>
</tr>
<tr>
<td>Root</td>
<td>Root, is</td>
</tr>
<tr>
<td>Det</td>
<td>Name, the</td>
</tr>
<tr>
<td>Nsubj</td>
<td>is name</td>
</tr>
<tr>
<td>Det</td>
<td>Place, the</td>
</tr>
<tr>
<td>Prep_of</td>
<td>Name, place</td>
</tr>
<tr>
<td>advmod</td>
<td>Born, where</td>
</tr>
<tr>
<td>Nsubjpass</td>
<td>Born, Gandhi</td>
</tr>
<tr>
<td>Auxpass</td>
<td>Born, was</td>
</tr>
<tr>
<td>Rmod</td>
<td>Place, born</td>
</tr>
</tbody>
</table>

**Table 2: Dependency Table for Question 1**

Pair word(x, y) returns the word paired with tag type x and word y.

For what type question:
- PW = pair word (attr, what) = is
- PW = pair word (! root, is) = name
- Since PW = name
- PW = pair word (some tag-type, PW) = place

Classifier = place

**Type of answer expected** (as derived from hypernym of place from word net) = location

**Question 2: Where was Mahatma Gandhi born?**

Pair word(x, y) returns the word paired with tag type x and word y.

For where type question:
- PW = pair word (Advmod, where) = Born
- PW = pair word (! (root —Aux), PW) = Gandhi
- Since PW = name
- Classifier = Gandhi

**Type of answer expected** (as derived from wh-word given in question (where)) = location

From the above example we can see that even though the same query was phrased in two different forms, the same answer type was returned.

2.3 Answer Analysis

Once we have analyzed the query and identified the expected answer type from the question, we can now find the precise
sections in the corpus that contain the answer to the question. The answering algorithm consists of two parts:
1) Preprocessing: Which involves paragraph ranking using biased entity-verb matching, and then generating the State Graph of the text corpus.
2) Answer Extraction: Which involves traversing the graph to reach the best answer using heuristic.

2.3.1 Paragraph Ranking
The vital nouns and verbs (primary keywords) of the questions are extracted (i.e. Verbs like was, did etc. or other common verbs or helping verbs are ignored if they are not the only verb of the question), and all paragraphs that contain the same nouns and verbs as those present in the query are ranked. Also, preference is given to the paragraphs of the corpus having the nouns and verbs of question occurring in the same sentence. This way, each paragraph is given a score which considerably reduces the overall text to be searched.

2.3.2 State Graph Generation
The ranked paragraphs are parsed using the Stanford Parser [11]. The paragraph with the best score is taken and parsed sentence by sentence to generate the state graph from the formulated rules. The basic format of the state graph is described in the fig 3.

Rules for Graph Generation:
- Proper Noun followed by Proper Noun is merged into a single noun node.
- Adjectives become a characteristic of the noun following it.
- If a verb is followed by several nouns separated by comma or conjunctions, then same verb is used as an edge from the noun previous to the verb (considered subject) to all the nouns (considered objects) following the verb.
- If more than one verb is present then there are multiple verb edges between two noun nodes.
- Same node is present as both subject and object noun for a verb, then there is no need to make that transition in graph.
- If there is a cause effect relationship between two nodes, then it is represented by temporal edge ci for cause numbered i and ei for effect numbered i. To mark a cause-effect pair: a list is maintained of all the words that indicate a cause-effect pair and also contain the data about the order of cause-effect i.e. whether cause follows effect or effect follows cause.
- E.g. Word = Therefore order = cause effect
  Word = Because order = effect cause
- Make all cardinal numbers as noun nodes.
- Sentences with no object noun have their subject connected to a Sink node by the verb.

The detailed example stating the state graph generation obtained from the paragraph using the above rules is described below. Example:
Mahatma Gandhi was born in the house of a senior government official Karamchand Gandhi. Karamchand Gandhi was married to Putlibai. Putlibai raised Mahatma Gandhi in coastal Gujarat. Mahatma Gandhi sought to practice nonviolence and truth. Mahatma Gandhi was trained in law in London. Mahatma Gandhi was too shy to speak up in court. Therefore Mahatma Gandhi attempts at establishing a law practice failed. Mahatma Gandhi fought and rebelled against White men.

Fig. 4 shows the state graph for the passage.

2.3.3 Graph Searching
The expected answer type is searched in the graph starting from the nouns of the question in a direction of the verb.
present in the question using a heuristic approach. A verb matrix is maintained which contain the verb, sentence number in which that verb is present, connecting nodes of the verb and cause_effect relationship if present between the connecting nodes. This verb matrix is examined and the sentence numbers having the same verbs as the question are marked.

Pseudo code for answer analysis is presented in fig. 5.

2.3.4 For cause_effect relationships
A node is selected such that its noun name matches with the noun of the question and it is also connected directly to the verb of the question. The closest effect node to it is selected (ei). Now the cause of that effect (ci) is selected. If this cause ci is also marked as the effect ej of some other cause cj, the cause leading to it is selected, till all the causes leading to the final event are selected. This way even the cause-effects, that are related, but are sentences apart are considered. The transitive relationships of cause effects are also considered. For example: if a causes b; b causes c; c causes d; d causes e.

The question What causes e? will give answer: ‘a’‘b’‘c’‘d’.

3. Evaluation
The evaluation has been performed on the TREC 2010 Dataset of about 1000 documents with 400 questions. The correct answer to each of the questions is known. Therefore, this dataset is used as a benchmark to test the proposed approach. The accuracy of the proposed method is judged using the Mean Reciprocal Ranking (MRR) as described below.

Mean Reciprocal Ranking
The accuracy is calculated by considering the first n answers for every question, where the values of n taken into account are like 1, 5, 10 and 30. The MRR of answer is calculated by the following expression:

\[
MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i}
\]

(1)

Where N is the number of questions and ranki is the rank of the correct answer for the ith question. If the question is not correctly answered in n attempts, then the reciprocal, \(1/rank_i\), is equal to 0. In the ideal case, the MRR should be equal to 1 i.e. all the questions are correctly answered by the top ranked answer. The worst case is when MRR is equal to zero i.e. the system could not find the correct answer to any of the questions in the top n results.

Figure 5 shows the results obtained by the proposed method when n = 5. 125 questions were correctly answered by the top ranked result resulting in a total score of 125 for MRR. Similarly, 171 questions were correctly answered by the second ranked response resulting in a contribution of 85.5 to MRR. There were 11 questions whose answer could not be found in the top five results. The overall MRR was 0.599.

It may be noted that if there is increment in the value of n then the MRR value will rise. This is because there is a small, positive contribution from the last 11 questions
whose correct answer could not be found in the top 5 results. The contributions of the other questions will not change by changing the value of n.

**First Hit Success (FHS)**

Another metric used for accuracy evaluation is considered by taking only the first answer for every question for the N questions fired. It is used where users solely depends on the system for finding the answer. If the first answer retrieved is correct, the FHS is 1 else it is 0. For the proposed work the FHS percentage = 31.25 %.

The question answering system based on semantic reformulation (M Ramprasath et al) which formulate the user query with the help of the argument extracted from the query and answer pair and retrieves the answer from the relevant documents extracted using web search engine had a precision of 0.588 and they havent considered the why type of question. On the other hand, this system has efficiency by 0.599(MRR) as well as this is capable to deal with both factoid and non-factoid type of questions.

### 4. Conclusion

This present work introduces a novel approach for extracting the answer to a question from a corpus data even when the questions are allowed to be unconstrained. The system works with an efficiency of 0.599 (MRR) value. The question analyzer gives the same expected answer type even when the same question is rephrased in complex forms. Also the answer analyzer extracts the answer distributed across various sentences, not necessarily occurring together in the paragraph or directly stated in the text.

### References


