Minimal parsing Key Concept based Question Answering System

Sunil Kopparapu, Sumitra Das, Akhilesh Srivastava, PVS Rao

July 17, 2006

Abstract

Question Answering (QA) systems are being increasingly used for information retrieval in several areas. QA systems are being proposed as 'intelligent' search engine that can act on a natural language query in lieu of the key word based search engines. In general, QA systems accept queries in natural language extract the intent of the query by parsing the query and then 'search' the database. One of the general requirements of a QA system is to act gracefully by providing 'close' answers from the database, in the absence of 'exact' answers, being present in the database. Most of the QA systems, today, parse the query to understand the intent of the query. In this paper we discuss why a full parsing might turn out to be a disadvantage and hence not necessary for a practical QA system. We propose a QA system based on minimal parsing of the query. We give examples from our working system to demonstrate the capability of the QA system.

1 Introduction

Question Answering (QA) systems are being increasingly used for information retrieval in several areas. QA systems are being proposed as 'intelligent' search engine that can act on a natural language query in lieu of the plain key word based search engines. While there are several QA systems in existence today, the common goal of most Question Answering (QA) systems is to (a) understand the query in natural language and (b) get a correct or an approximately correct answer in response to a query from a predefined database.

In a very broad sense, QA system can be thought of as being a pattern matching system. The query in its original form (as framed by the user) is preprocessed and parameterized and made available to the system in a form that can be used to match the answer paragraphs. It is, however, assumed that the answers have also been preprocessed and parameterized in a similar fashion. The extracted parameters could be as simple as picking selective key words and/or key phrases from the query and then matching these with the selected key words and phrases extracted from the answer paragraphs. On the other hand it could be as complex as fully parsing the query, to extent of identifying the parts of speech of each word in the query, and then matching the parsed information with full parsed answer paragraphs. The preprocessing required would generally depend on the type of parameters being extracted. For instance, for a simple key words type parameter extraction, the preprocessing would involve removal of all words that are not key words (example, punctuation, connecting words, adjectives etc) while for a fully parsing system it could be retaining the punctuations and verifying the syntactic and semantic accuracy of the query.

Most QA systems take the approach of full parsing to comprehend the query. While this is good (able to determine who killed whom in a sentence "Rama killed Ravana") it is far from satisfactory in performance in practice because for accurate and consistent parsing the following is required

- The implemented parser, used by the QA system should know the rules of grammar
- Person writing the sentence (both query and answer paragraph) should know the rules of grammar

If either of the conditions fails, the QA system will not perform to satisfaction. While one can take care of the parser to follow the rules of grammar, it is far more constraining (impractical) to accept the same degree of grammar competence from a casual user of the system. Unless we make sure that the person writing the sentence is grammatically correct - the flawless parser could run into problem. For example

- a full sentence parser would take a grammatically incorrect constructed query from a user and give a sense to the whole sentence query (assuming that the query is grammatically correct). The analysis and hence the interpretation of the query by the system could be far from the exact intent of the query.
- Parsing need not necessarily gives the correct or intended result. The following sentence, "Time flies like an arrow", is a well known example, which can be parsed in several ways, namely,
  - time flies (funny critters, probably out of Star Trek or something) like an arrow?
  - or does it mean, time (as in space-time) flies like an arrow?
  - does it mean, time (as a verb) flies like an arrow? (time the flies, as if they were an arrow.

Full parsing is very difficult and we believe that it is not required in general for a QA system especially because we envisage the use of the QA system by

- large number of people who need not be necessarily grammatically correct all the time,
- people would wish to use casual/verbal grammar (intent is conveyed but from a purist angle the sentence construct is not correct).

Our approach is one of middle path, neither too simple not too complex.

2 Our Approach

The goal of our QA system is (i) to get a correct or an approximate correct answer in response to a query and (ii) not put a constraint on the user to construct syntactically perfect queries[1]. This dictates that we choose an approach which is mid path - neither too simple not too complex. There is no one strategy envisaged - a combination of strategies based on heuristics, we believe work best for a practical QA system. The proposed QA system uses a middle

[1]In general, verbal communication (especially if one thinks of a speech interface to the QA system) uses informal grammar and most of the QA systems which use full parsing would fail.

path especially because the first approach (picking up key words) is very simplistic and could result in a high false answer fetching (false acceptance), the second approach, namely, the full parsing approach is complex, time consuming and could end up rejecting valid answers (false rejection), especially if the query is not well formed syntactically. The system is based on two types of parameters – key words (KW) and key concepts (KC).

Key Word (KW) and Key Concept (KC): A KW is an important word (neither too frequently occurring nor too rarely occurring!) that by themselves makes one paragraph different from other (Note that pronouns, connecting words, common words are not KWs because by themselves they don't differentiate one paragraph from another) and a KC is an extra important key word that holds the paragraph together[2]. Very loosely speaking a KC can be defined as an extra important KW.

For example, given the paragraph

The data is now on a new ftp server ftp://81.10.29.101/root/Tata/. With the same user name and password. I'll also send you the files on your gmail account.

The KW's in this paragraph are data, ftp_server, ftp://81.10.29.101/root/Tata/, files, user_name, password, gmail_account while the KC is send. Key Words are generally

- not general,
- contain extra information,
- have high entropy,
- help make more specific the choice of answers,
- contain information which helps in narrowing the domain of search.

Key Concept has several interpretations. It can be considered as that word that hold the key to the central idea of a paragraph; if we changed the KC, most likely the central idea of the paragraph changes distinctly. Consider the following

Could I reserve a ticket to go to Bangalore from Mumbai by 2nd class in the first week of March in a train that start in the morning and reaches Bangalore before midnight the same day?

The KC here is reserve. Change the KC to, say, cancel and the whole meaning of the paragraph changes. Incidentally, the KW are ticket, Bangalore, Mumbai, 2nd class, first_week, March etc ...

The KC can be looked upon as a mathematical functional[3] which relates other words (mostly KWs) to itself; if you change this functional word (KC), the total meaning of the whole paragraph is bound to change. Representational example,

$$KC_1(KW_1, KW_2, KC_2(KW_3, KW_4))$$

In most cases verbs[4] in a query serve as KC because they hold the paragraph together. Examples reserve, activity, plan, manage etc

[2]in many cases it is a verb
[3]a function of a function
[4]part of speech

In our approach it is essential to identify KC because it helps in identifying the intent of the query rather; which we believe is crucial for an intelligent QA system. This ability gives it an edge over the simplistic QA system based on KWs only. Identifying KCs helps in better understanding the query and hence the system is able to answer the query more appropriately. A query in all likelihood will have but one KC but this need not be true with the KCs in the paragraph. If more than one key concepts are present in a paragraph, one talks of hierarchy of key concepts, meaning which if the several KCs present in the paragraph is more important than the other (unless otherwise specified, we will assume that there is only one KC in an answer paragraph).

KCs together with KWs help in capturing the total intent of the query. This results in constraining the search[5] and making the query very specific. For example,

$$reserve(place\_from = Mumbai, placeto = Bangalore, class = 2nd)$$

makes the query more specific (exact) - ruling out the possibility of a reservation between Mumbai and Bangalore in 3rd AC for instance. In general, KC captures the intent and the KWs make the search domain more specific and help determine what specific information to seek or give.

One can think of a QA system based on key concept and key words as one that would save the need to fully parse the query; this comes with a cost, namely, this could result in the system not being able to distinguish who killed whom in the sentence ”Rama killed Ravana”. A full parsing system would be able to determine that Ravana is the victim and and Rama the killer. While our, KC-KW based QA system it would be represented as as kill (Rama, Ravana) which can have two interpretations.

But in general, this is not a huge issue unless there are two different paragraphs - the first paragraph describing about Rama killing Ravana and a second paragraph (very unlikely) describing Ravana killing Rama.

3 Conclusions

Our QA system is designed to work optimally for even informal grammar (verbal/casual communication as against formal communication) and hence doesn't have a full sentence parser. The absence of a full parser is not a constraint because in a limited domain the context is not ambiguous. A simplistic view of our approach can be summed as the process of template filling. The KC help in identifying the template while the KW fill in the slots. Like UNL[6], linkages between words, exist implicitly in our approach which are based on the dimensionality of the word of the KW.

[5]Pattern to be searched

[6]Universal Networking Language. Uses linkages between words in a sentence to identify the intent of the query.