KisanMitra: A Question Answering System For Rural Indian Farmers

Sunil Kopparapu, Akhilesh Srivastava, PVS Rao
Cognitive Systems Research Laboratory, Tata Infotech Limited,
Plot No 14, Sector 24, Vashi-Turbe, Navi Mumbai 400 705, India.
{SunilKumar.Kopparapu, Akhilesh.Srivastava,PVS.Rao}@TataInfotech.Com

***Abstract*— Farmers in most rural areas in India not only need expert and timely suggestion to obtain rich harvest of their crops but also need information regarding the subsidies, government schemes to make cultivation pay rich dividends. Expert guidance comes in the form of an human expert visiting the village and the farmers being able to get their turn to seek answers to their queries. In this paper, we propose a Question Answering (QA) system, which would act as an expert and answer queries of the farmers. We call this QA system KisanMitra, friend of the farmer. The idea in building this system is to give access to information 24×7, to keep the information that reaches the farmer updated, enable the farmer to query in his own language without being strict on grammar or construct of the query. The system is intelligent in the sense, it understand the intent of the query and provides responses. In the absence of exact answers not being present in its KisanMitra, it provides answers which are close in some sense.**

I. Introduction

Search engines such as AltaVista [1] and Google [2] typically treat a natural language question as a list of keywords and retrieve documents that have *similar* keywords. However, documents with the best answers may not contain all the keywords in the original query and hence may be ranked low by the search engine. These queries could be answered more precisely if a search engine recognized them as questions as humans would do and not merely treat the queries as a set of keywords [3]. To cater to the growing world wide web and the need to search by the semantic intent of the query a large number of Question Answering (QA) systems have been researched, eg [4], [5], [6].

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 *KisanMitra*. One of the general requirements of a QA system is to act gracefully by providing *close* answers from the *KisanMitra*, in the absence of *exact* answers, being present in the *KisanMitra*. Question Answering (QA) systems are being increasingly used to provide a better and more natural user interface (UI). Research in the area of QA systems is to make them intelligent so that they accept queries in natural language, understand the intent of the query and then respond quickly with answers which are useful and apt.

This paper describes a closed domain QA system, *KisanMitra*, which has been developed keeping the rural farmers as the main users. The system is built on a Natural Language processing engine researched and developed by Tata Infotech[7]. *KisanMitra* addresses various issues concerning the farmer in rural India. The QA system is aimed to act like an expert, in the absence of an human expert, giving information to the queries posed by farmers. Currently the system is capable of answering queries related to fertilizers, pesticides and government schemes.

KisanMitra is a web enabled system (Fig. 1), with an Hindi interface. The system is capable of understanding *even* grammatically incorrect constructs of a sentence in addition to being being capable of correcting spelling mistakes. This facilitates the user to pose their query without any constraints. This power of *KisanMitra* allows the user to query the system in the same form as (s)he would articulate (with out constraint) his query to another human to get a response.

II. Question Answering Systems

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

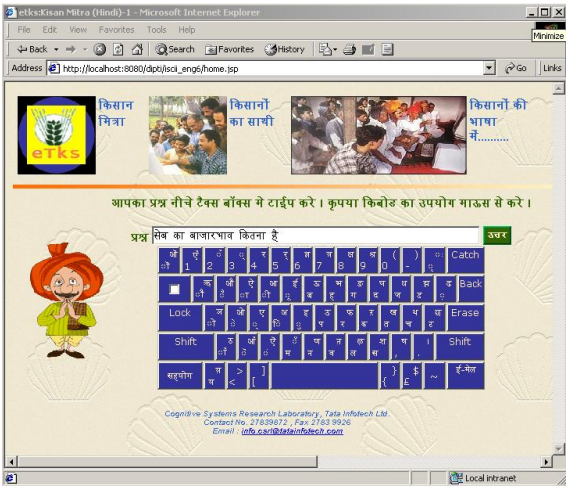

Fig. 1. *KisanMitra* front end. Hindi keyboard is optional and is provided to facilitate easy input in Hindi.

in the knowledge-base. 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 in the knowledge-base. On the other hand it could be as complex as fully parsing the query, to the extent of identifying the parts of speech of each word in the query, and then matching the parsed information with fully parsed answer paragraphs. The preprocessing required would generally depend on the type of parameters being extracted. For instance, in a simple key word type parameter extraction, the preprocessing would involve removal of all words that are stop words (example, punctuation marks, connecting words, pronouns, adjectives etc) while for a full 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 like *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 the answer paragraph) should know the rules of grammar

If either of the conditions fail, 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 querying is grammatically correct the parser could run into problem. For example, a full sentence parser would take a grammatically incorrect constructed query from a user and give a *different* 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 *actual* intent of the query. In addition, parsing need not necessarily give 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?
- time (as in space-time) flies like an arrow?
- time (as a verb) flies like an arrow? (time the flies, as if they were an arrow.

Full parsing is very difficult, in our context because of the targeted user. Further, we believe that full parsing of a query, in general, is not required for a QA system. This is because we envisage the use of QA systems 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 (key words based) nor too complex (full parsing based).

III. 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 any 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 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,

[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.

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).

A. Key word (kw) and key concept (kc)

A kw is an important word[2] that by themselves makes one paragraph different from other (Note that pronouns, connecting words, common words are not kw's because by themselves they do not differentiate one paragraph from another) and a kc is an extra important kw that holds the paragraph together[3]. Key Concept has several interpretations

- it holds the key to the central idea of a paragraph; if we changed the kc most likely the central idea of the paragraph changes distinctly.
- a mathematical functional (a function of a function) which relates other words (mostly kw and sub key concepts) to itself; if you change this functional word (kc), the total meaning of the whole paragraph is bound to change. Example, $kc_1(kw_1, kw_2, kc_2(kw_3, kw_4))$
- In most cases verbs (part of speech) serve as key concept because they hold the paragraph together. Examples reserve, activity, plan, manage etc

Very loosely speaking a kc can be defined as an extra important kw.

IV. Implementation

KisanMitra is a web based system sitting on a server and is accessible on the http protocol (Fig. 2 and can be accessed using a web browser (eg. Netscape, IE, mozilla etc). The server end architecture of *KisanMitra* essentially consists of 4 major modules. The *input interface* module takes the query in an Indian language (currently in Hindi, English) and preprocesses it before passing it on to the *question understanding* (QU) modules. The preprocessing step involves removing stop words and extraneous characters like punctuation marks. Additionally, using a lookup the preprocessor is able top show the user a list of possible alternatives for a misspelt word for the user to choose from.

QU module is by and large the most important module of the QA system. This module extracts

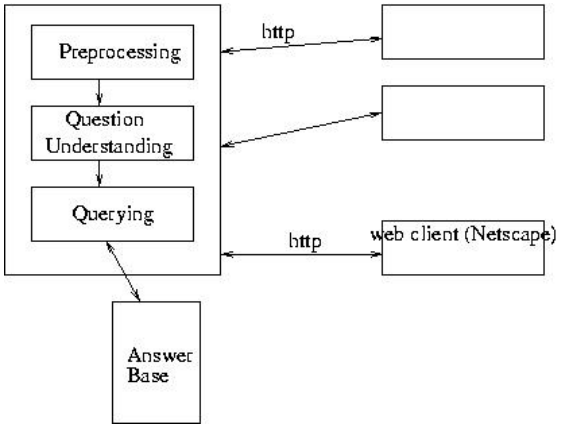

Fig. 2. Block diagram of *KisanMitra*.

kc's and kw's from the preprocessed query making use of the taxonomy three (also called ontology). The taxonomy is generally constructed or engineered manually for a given domain. Typically ontology can be though of as a special kind of a dictionary that holds the relationship between different words. Additionally, each word in the taxonomy has a kw, kc tag. These tags are generally derived through statistical analysis of the domain data for which the QA is being built. The QU module constructs the pattern, $kc(kw_1, kw_2)$, by analyzing the query[4].

The pattern is then used by the *query module* to query the knowledge-base. If an exact pattern is found in the knowledge-base, the corresponding answer is displayed to the user. In the absence of an exact answer being found, another pattern $kc(kw_1', kw_2)$[5] is generated using the taxonomy tree. Clearly, the relevance of the answer fetched for $kc(kw_1', kw_2)$ is less than the answer fetched for $kc(kw_1, kw_2)$, but since kw_1' and kw are related, the answer fetched by $kc(kw_1', kw_2)$ is in general *useful*. For example, for the query, *How can get rid of X which is destroying my moong dal*, the QU system would generate $get\_rid\_of(X, moong\ \ dal)$ and if not answer were found then it would generate $get\_rid\_of(Y, moong\ \ dal)$ followed by $get\_rid\_of(X, pulses)$ and $get\_rid\_of(Y, pulses)$. Clearly all these patterns fetch answers that are close in some sense to the query posed.

[2]neither too frequently occurring nor too rarely occurring!

[3]in many cases it is a verb, but not necessarily

[4]In general every query has a single kc; but in the unlikely event of the system identifying more than one kc - the QU module generates $kc_1(kw_1, kw_2)$, $\cdots$ $kc_n(kw_1, kw_2)$ etc ..

[5]kw_1' and kw are related in some sense in the taxonomy tree.

V. Conclusions

Question Answering system are attractive and are being extensively used in many applications. QA systems enable a user to get to information quickly without several hyperlink clicks and manual search that are typically required to get to the required information on a web page. In this paper, we have described a QA system that specifically addresses the rural population. The strengths and the unique features of the system (*KisanMitra*) is its ability to (i) take queries in Indian language through a Hindi soft keyboard, (ii) accept query in natural language, which need not be grammatically correct and (iii) the system responds to even queries which do not have an exact answer in its knowledge-base by supplying answers close to the query. These features make the system user friendly and provide a more natural interface for the user to query the system. A system prototype has been built and is in the process of being tested for deployment to the rural masses.

References

[1] AltaVista, in *http://www.altavista.com*, website.

[2] Google, in *http://www.google.com*, website.

[3] E. Agichtein, S. Lawrence, and L. Gravano, "Learning search engine specific query transformations for question answering," in *Proceedings of the Tenth International World Wide Web Conference*, 2001.

[4] AskJeevs, in *http://www.ask.com*, website.

[5] AnswerBug, in *http://www.answerbug.com*, website.

[6] START, in *http://start.csail.mit.edu/*, website.

[7] Tata Infotech Limited, in *http://www.tatainfotech.com*, website.