Question Answering System

Shekhar Nalawade[1],Rupali Palve[2], Vikram Kumar[3]
[1,2,3]*Computer Department, Pune University*
Sinhgad Technical Education Society, Department of Computer Engg, Lonavala, Maharashtra, India
[1]smn.sit@sinhgad.edu, [2]palverup@gmail.com, [3]vkrmsingh023@gmail.com

***Abstract*—As the amount of information available online started to grow exponentially, the need for increasingly sophisticated search tools led to the creation of Web search engines that allowed users to retrieve documents based on keyword queries. While search engines are very well suited for retrieving relevant documents, they are much less effective when users need to find very specific pieces of information.**

To reduce time and effort in formulating effective queries, question answering (QA) systems were proposed as an alternative to Web search engines to help users who need to find small pieces of factual information rather than whole documents. Question answering is a specialized type of information access in which systems must return an exact answer to a natural language question. It uses natural language processing (NLP) techniques to process a question, then searches for the required information to identify the answer and presents the answer to the user.

Our proposed solution of QA Systems works on specific domain of tourism where it is enriched to answer many questions related to tourism like distance,hotels,resort etc..,

Keywords— Information retrieval, databases, crawler, Tokens, Seed URL.

I. Introduction

Traditionally, question answering on the Worldwide Web (WWW) is done by the means of static lists of frequently asked questions (FAQs) and their answers. In such a list, people locate the questions of their interest and read the answers. This practice, however, has two shortcomings. First, the readers of an FAQ list do not explicitly ask any questions, therefore the information provider does not know all the variety of questions that arise. Second, finding valuable information in a long – several hundreds of entries – and chaotic FAQ list, or a number of lists, is a tedious work.

Keyword-based search, used by most search engines, is a common means of document retrieval on the Web. Many of us, however, do not think in terms of Boolean expressions made of keywords and are not used to the engine-specific syntax of such expressions. Another inconvenience of the keyword queries is the large amount of retrieved irrelevant information.

Input forms are common user interfaces for structured (e.g., SQL) databases. Input forms are convenient if they are small, but tedious if there are many input fields.

Supposedly, the most natural kind of queries posted to an information system is questions stated in ordinary human language. In a question, the user can specify exactly what he or she wants. It is more fun to talk to a computer in ordinary English. A natural language based interface does indirect interviewing of the users: in the logs of the system we read what people think when they search for information. On the web such an interface adds one more dimension – limited human language understanding – to the traditional notion of multi-media (images, sounds, animation).

The World Wide Web has grown dramatically since its inception in 1992 as a global interconnected system for document sharing amongst researchers. With over 130 million domains and a billion unique URLs and with more than two billion estimated users, it has fundamentally transformed the way information is shared, distributed and accessed.

As users struggle to navigate the wealth of on-line information now available, the need for automated question answering systems becomes more urgent. We need systems that allow a user to ask a question in everyday language and receive an answer quickly and succinctly, with sufficient context to validate the answer. Current search engines can return ranked lists of documents, but they do not deliver answers to the user.

Question answering systems address this problem. Recent successes have been reported in a series of question-answering evaluations that started in 1999 as part of the Text Retrieval Conference (TREC). The best systems are now able to answer more than two thirds of factual questions in this evaluation. The combination of user demand and promising results have stimulated international interest and activity in question answering. This special issue arises from an invitation to the research community to discuss the performance, requirements, uses, and challenges of question answering systems.

To answer a question, a system must analyze the question, perhaps in the context of some ongoing interaction; it must find one or more answers by consulting on-line resources; and it must present the answer to the user in some appropriate form, perhaps associated with justification or supporting materials.

This section provides an overview of some dimensions of this research in terms of:

- Questions
- Answers
- Evaluation
- Presentation

Questions:

We can distinguish different kinds of questions: yes/no questions, “wh" questions (who established mahabaleshwar, what is the distance between pune and lonalva), indirect requests (I would like you to list ...), and commands (Name all the hotels...). All of these should be treated as questions. However, systems that depend heavily on the use of “wh" words for clues (who needs a person answer, when needs a time answer) may have difficulty processing such questions.

We have evidence that some kinds of questions are harder than others. For example, why and how questions tend to be more difficult, because they require understanding causality or instrumental relations, and these are typically expressed as clauses or separate sentences .If a system does a good job of analyzing the type of answer expected, this narrows the space of possible answers. Certain kinds of questions are harder to answer because of an insufficiently narrowed answer type; for example, what questions are notoriously hard, because they provide little constraint on the answer type.

Answers:

Answers may be long or short, they may be lists or narrative. They may vary with intended use and intended user. For example, if a user wants justification, this requires a longer answer. But short answer reading comprehension tests require short answers (phrases).

There are also different methodologies for constructing an answer: through extraction - cutting and pasting snippets from the original document(s) containing the answer - or via generation. Where the answer is drawn from multiple sentences or multiple documents, the coherence of an extracted answer may be reduced, requiring generation to synthesize the pieces into a coherent whole.

Evaluation:

What makes an answer good? Is a good answer long, containing sufficient context to justify its selection as an answer? Context is useful if the system presents multiple candidate answers, because it allows the user to find a correct answer, even when that answer is not the top ranked answer. However, in other cases, short answers may be better. The experiences of the TREC question answering evaluations [1] show that it is easier to provide longer segments that contain an embedded answer than shorter segments. In section 4, we discuss issues of evaluation and criteria for question selection and answer correctness in greater detail.

Presentation:

Finally, in real information seeking situations, there is a user who interacts with a system in real time. The user often starts with a general (and underspecified) question, and the system provides feedback directly or indirectly by returning too many documents. The user then narrows the search, thus engaging in a kind of dialogue with the system. Facilitating such dialogue interactions would likely increase both usability and user satisfaction. In addition, if interfaces were able to handle both speech input and dialogue, question answering systems could be used to provide conversational access to Web based information - an area of great commercial interest, particularly to telecommunications and Web content providers.

To date, there has been little work on interfaces for question answering. There have been few systematic evaluations of how to best present the information to the user, how many answers to present to a user, how much context to provide, or whether to provide complete answers vs. short answers with an attached summary or pointers, etc. This is an area that will receive increased attention as commercial question answering interfaces begin to be deployed.

In our proposed system we developed a QA System for tourism domain. Where by using a crawler we collected parsed web page content of many tourism site web pages and then preprocess the web information by tokenization, Stop word removing and stemming to store them in file systems.

Then auto token System will take all these file information and extract the key words and store with its belonged File URL in the database or this can be done manually and save in database actually this configures token file.

Related answer strings are specified and saved in Database.Like km, miles, meters, Yards are the words for any distance related Questions. And then link these with the tokens for finding answer in the collected information.

When user fires query to our QA system then query is preprocess by tokenization, Stop word removing and stemming and then Query is fed to the token file object to identify the type of answer (for Example: If the question contains keyword like distance then the answer should contain words like km,kilo meters, miles etc..).Then based on the procedure programming style for specific key word our system searches the answer by performing natural language processing.

The rest of the paper is organized as follows. Section 2 discusses some related work and section 3 presents the design of our approach. The details of the results and some discussions we have conducted on this approach are presented in section 4 as Results and Discussions. Sections 5 provides hints of some extension of our approach as future work and conclusion.

II. Related work

The development of systems that interact with human users in natural language has long been a goal of the artificial intelligence research community. Since the 1960s, when the field was in its infancy, a variety of natural language database front-ends, dialog systems, and language understanding systems have been created.

Current QA Systems are capable of evaluating answers from complex system of data. Many of the current QA systems are for closed domains, that is, specific topic such as medical topics, or for limited types of questions only, such as descriptive questions. The problem with the current QA system is that they suffer from low recall. The answer to question is also limited to predefined categories [1].

Restricted-domain QA has a long history, beginning with systems working over databases. e.g., BASEBALL [2] and LUNAR [3],But as we reviewed above, the current trend in Question Answering focus on open domain, which has been largely driven by the TREC-QA Track. Nonetheless, QA system of open domain is lacking to treat the special domains for all question types, because no restriction is imposed either on the question type or on the user's special vocabulary and it is very hard to construct a common knowledge (ontology) base for open domain.

Unlike Artificial Intelligence question answering systems that focus on generation of new answers, FAQ answering systems retrieve existing answers from their databases. Auto-FAQ [4] and FAQ Finder [5] are two representative systems aimed at automating navigation through FAQ sets. They have three common core features:

- The systems use a natural language based interface – a user asks his or her question in ordinary English.
- The systems answer it by one or several pre-stored related questions and their answers, if any.
- Both systems interact with their users through WWW (initially FAQ Finder did not have a Web-based user interface).

The high accuracy in answer extraction has been greatly achieved by using heuristics. [6] also fully parses questions and then apply a large number of rules to the parse tree to classify questions. In contrast, a machine learning approach can automatically construct a high performance question classification program which leverages thousands or more features of questions.

Given more training data, the performance of a learned classification program usually improves. Moreover, a learned classification program is more flexible than a manual one since it can be easily adapted to a new domain. And there are some papers describing machine learning approaches to question classification, such as [7] use support vector machines, a machine learning approach. [8] Uses language models for question classification.

Once the type of entity being sought has been identified, the remaining task of question analysis is to identify additional constraints that entities matching the type description must also meet. This process may be as simple as extracting keywords from the rest of the question to be used in matching against candidate answer-bearing sentences. This set of keywords may then be expanded, using synonyms and/or morphological variants[9] or using full-blown query expansion techniques by, for example, issuing a query based on the keywords against an encyclopedia and using top ranked retrieved passages to expand the keyword set [10]. Or the constraint identification process may involve parsing the question with grammars of varying sophistication.

[11] Use a wide-coverage statistical parser which aims to produce full parses. The constituent analysis of a question that it produces is transformed into a semantic representation which captures dependencies between terms in the question. [12] use a robust partial parser which aims to determine grammatical relations in the question where it can (e.g. main verb plus logical subjects and objects). Where these relations link to the entity identified as the sought entity, they are passed on as constraints to be taken into account during answer extraction.

This QA System is the one of highly enriched and inseparable part of information Retrieval (IR) System. There are many types of IR system protocols are been using in the present day scenario's, like

Vector Space Model (VSM): This is first introduce by [13], models both the documents in the collection and the query strings as vectors in a finite dimensional Euclidean vector space.

Probability Retrieval Models: The initial idea of probabilistic retrieval was proposed by Maron and Kuhns in a paper published in 1960 [14].and it is based on probability that the document is relevant to the query.

Inference Network Model: In this model, document retrieval is modeled as an inference process in an inference network [15]. Most techniques used by IR systems can be implemented under this model.

Information extraction (IE) is a new technology enabling relevant content to be extracted from textual information available electronically. IE essentially builds on natural language processing and computational linguistics, but it is also closely related to the well established area of information retrieval and it is as a method of searching for information in some ways similar to Question Answering. Generally,

The process of IE has two major parts. First, the system extracts individual "facts" from the text of a document through local text analysis. Second, it integrates these facts, producing larger facts or new facts (through Inference). As a final step after the facts are integrated, the pertinent facts are translated into the required output format.

There are many IE systems are been proposed and using in the research area. Some of the IE types are discussed below in brief.

Template Matching: Formerly known as message understanding, the general goal of information extraction is to locate information within free text that matches prepared templates.

Named Entity Recognition **:**Named Entity (NE) Recognition is a specialized form of the IE task dedicated to identifying phrases in text that refer to entities like people, organizations, date, dates and currency amounts and facilities, and extracting their semantics.

Automated Content Extraction (ACE) is a large-scale evaluation effort for IE systems run by the National Institute of Standards and Technologies (NIST). ACE challenges participating systems to locate references of people, geo-political entities such as cities, states and nations, locations with physical extent, organizations and facilities within newswire text and broadcast news transcripts.

III. PROPOSED METHOD

In this section, we describe our approach of Question Answering System with a heuristic approach for the steps shown in figure 1. As shown in figure there are 9 main steps in our approach.

Step 1*:* Here user enters a question through user interface in linguistic form.

Step 2: This is the step where we are preprocessing of user question is conducted, where query entered by the user is bring down to its basic meaning words by the following four main activities: Sentence Segmentation, Tokenization, Removing Stop Word, and Word Stemming.

Sentence segmentation is boundary detection and separating source text into sentence. Tokenization is separating the input query into individual words. Next, Removing Stop Words, stop words are the words which appear frequently in the query but provide less meaning in identifying the important content of the document such as 'a', 'an', 'the', etc.. The last step for preprocessing is Word Stemming; Word stemming is the process of removing prefixes and suffixes of each word.

Step 3: This is the key step to our answer extraction process, where we are identifying the tokens which are defining many of the possible domain question's answerable token keywords that enable our system to search question more efficiently. For example, for the *distance* related query answer always with its unit like *km, kilo meter, miles* etc...

Step 4: This step actually decides the quality of the answers providing by our system. Here we select many of the tourism domain website where information is been properly defined. For our approach we consider web pages of tourism places around pune city of Maharashtra state, India.

Step 5: In this step we are creating a web crawler which accepts a seed URL of tourism domain and searches it's all links.

Web crawlers are an essential component to search engines; running a web crawler is a challenging task. There are tricky performance and reliability issues and even more importantly, there are social issues. Crawling is the most fragile application since it involves interacting with hundreds of thousands of web servers and various name servers, which are all beyond the control of the system.

Web crawling speed is governed not only by the speed of one's own Internet connection, but also by the speed of the sites that are to be crawled. Especially if one is a crawling site from multiple servers, the total crawling time can be significantly reduced, if many downloads are done in parallel.

Despite the numerous applications for Web crawlers, at the core they are all fundamentally the same. Following is the process by which Web crawlers work:

- Download the Web page.
- Parse through the downloaded page and retrieve all the links.
- For each link retrieved, repeat the process.

The Web crawler can be used for crawling through a whole site on the Inter-/Intranet. When we specify a seed URL and the Crawler follows all links found in that HTML page. This usually leads to more links, which will be followed again, and so on. A site can be seen as a tree-structure, the root is the seed URL; all links in that root-HTML-page are direct sons of the root. Subsequent links are then sons of the previous sons [17] [18].

Here in our proposed method we developed a web crawler using java programming language, where we used multithreading feature extensively and also used java html parser to parse the web pages. And finally we store all collected web links in the database.

Step 6: This is the one of the most crucial phase of our experiment, where our system interact with the live web page of the tourism domain URL. And then by using a well designed baby web crawler our system is enable to fetch the data of the web page and then parse all the HTML tags from the web page. Only human readable data is extracted from the web page and also many advertisements contents are also vomited in this phase.

Step 7: The parsed data which is collected in the step 6 is again send to preprocessing method of step 2 to bring the data in very ease form and then this data is saved in a specific location in the file form.

Step 8: This step is the engine of our system, where tokens and query keywords are process to get the answer for the specific question.

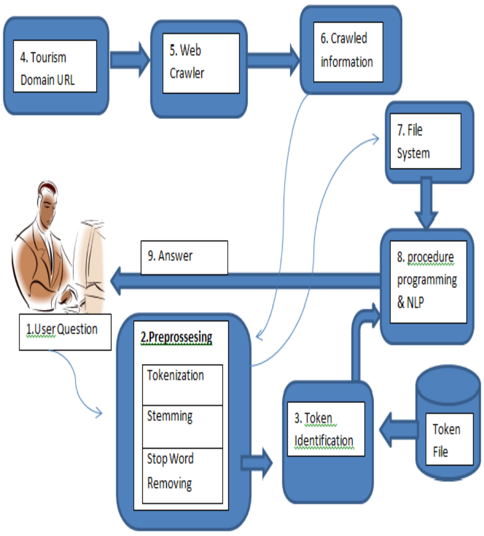

Figure1.Overview of system approach

Here a generalized steps are mentioned below which is followed by our system

- Construct a master vector which constitutes a set of token and keyword (like kilometer, distance)
- Extract the number of the sentences in the document
- For each sentence identify the master vector elements are found then label the sentence.
- Identify the noun in the sentence (here we used a dictionary file to do so).
- If there is more number of token words in a sentence then identify the nearest token to noun of the question.
- Segment the answer word and extract from the sentence.

These steps are representing as algorithm as below.

Algorithm 1 Our approach

// input: Question Q_n

//input: Dictionary Set $D_c = \{d_1, d2, d2 \ldots . d_n\}$

Where d_n is dictionary words

// output: Answer A_n

1: Set $M_v = \{T_k, Kw\}$ (Master vector, token, keyword)

2: **For** each sentence S_i i=1 to N

3. **If** $M_v \in S_i$ then

4. **tag** S_i as S_{imp}

5. $S_{imp} \neq D_c \rightarrow P_n$ (Proper Noun)

6. (Words of S_{imp}) $W_i \rightarrow P_n \rightarrow A_n$

7. **return** A_n

Step 9: Here answer are collected as a single word or multiple then arrange them as a list and display to the user.

REFERENCES

[1] kim , H.,Kim,k.,Lee ,G.G., & Seo ,J.(2001).MAYA: "A Fast Question Answering system based on a predictive answer indexer."

[2] Green, W., Chomsky, Laugherty, C. K.(1961). BASEBALL: An automatic question answerer, Proceedings of the Western Joint Computer Conference, pp219- 224.

[3] Woods, W. A.(1973). Progress in natural language understanding: An application to lunar geology, AFIPS Conference Proceedings, Vol. 42, pp.441-450.

[4] Whitehead S. D. (1995) Auto-FAQ: an Experiment in Cyberspace Leveraging. Computer Networks and ISDN Systems, Vol. 28, No. 1-2, pp. 137-146.

[5] Hammond K., Burke R., Martin C., and Lytinen S. (1995) FAQ Finder: a Case-Based Approach to Knowledge Navigation. Proceedings. The 11th Conference on Artificial Intelligence for Applications, Los Alamitos, CA, USA, IEEE Comput. Soc. Press, pp. 80-86.

[6] Hermjakob, U.(2001). Parsing and question classification for question answering, In Proceedings of the Workshop on Open-Domain Question Answering at ACL-2001, Toulouse, France. July 6-11.

[7] Zhang, D. & Lee, W.(2003). Question classification using support vector machines, In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.26-32.

[8] Li, W.(2002). "Question classification using language modeling," Technical Report IR-259, Center for Intelligent Information Retrieval, University of Massachusetts.

[9] Srihari, R. & Li,W. (2000). "Information extraction supported question answering" In Proceedings 8th Text Retrieval Conference (TREC-8), NIST Special Publication 500-246.

[10] Ittycheriah, A., Franz, M., Zhu, W, J. & Ratnaparkhi, A. (2001). IBM's "statistical question answering system." In Proceedings 9th Text Retrieval Conference (TREC-9), NIST Special Publication 500-249.

[11] Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Rus, V. & Morarescu, P. (2000). FALCON: Boosting knowledge for Answer Engines, The Ninth Text REtrieval Conference (TREC 9), NIST Special Publication, 500-249.

[12] Scott, S. & Gaizauskas, R. (2001). University of Sheffield TREC-9 Q&A System. In Proceedings 9th Text Retrieval Conference (TREC-9), NIST Special Publication 500-249.

[13] Salton, G., Wong, A. & Yang, C. (1975). A vector space model for automatic indexing, Communications of the ACM, 18(11), pp.613-620.

[14] Maron, M. E. & Kuhns, J. L.(1960). On relevance, probabilistic indexing and information retrieval, Journal of the ACM, 7, pp.216-244.

[15] Turtle, H. R. (1991). Inference Networks for Document Retrieval, PhD thesis, CIIR, University of Massachusettes.

[16] Ellen, M. V.(2001). Overview of the TREC 2001 question answering track, *In Proceedings of the 2001 Text REtrievalConference (TREC 2001)*.

[18] Franklin, Curt. How Internet Search Engines Work, 2002. www.howstuffworks.com

[19] Grossan, B. "Search Engines: What they are, how they work, and practical suggestions for getting the most out of them," February1997. http://www.webreference.com