SOUTH ASIAN LANGUAGE REVIEW
VOL.XIII, Nos 1&2, January-June,2003.

A Query Answering System for E-Learning Hindi Documents

Praveen Kumar, Shrikant Kashyap, Ankush Mittal
Indian Institute of Technology, Roorkee, India
{pkmaxuec,shrikuec,ankumfec}@iitr.ernet.in

Sumit Gupta
MVGR College of Engineering, A. P., India
sumitfec@mail.com

Abstract. To empower the general mass through access to information and knowledge, organized efforts are being made to develop relevant content in local languages and provide local language capabilities to utility software. We have developed a Question Answering (QA) System for Hindi documents that would be relevant for masses using Hindi as primary language of education. The user should be able to access information from E-learning documents in a user friendly way, that is by questioning the system in their native language Hindi and the system will return the intended answer (also in Hindi) by searching in context from the repository of Hindi documents. The language constructs, query structure, common words, etc. are completely different in Hindi as compared to English. A novel strategy, in addition to conventional search and NLP techniques, was used to construct the Hindi QA system. The focus is on context based retrieval of information. For this purpose we implemented a Hindi search engine that works on locality-based similarity heuristics to retrieve relevant passages from the collection. It also incorporates language analysis modules like stemmer and morphological analyzer as well as self constructed lexical database of synonyms. The experimental results over corpus of two important domains of agriculture and science show effectiveness of our approach.

1. Introduction

With the web content being written in different languages of the world, it has become important to have tools that can retrieve information from the documents written in different languages. In the context of Indian languages, Hindi language has been given much emphasis leading to the development of significant number of Hindi documents. In fact, of the top 100 languages in the world, English occupies the top position, with Hindi coming fifth.

E-learning is a novel method for presenting information to students for the purpose of education. The government has envisioned providing primary and secondary education through E-learning. With computer education facilities being set up in secondary schools at block level throughout the country, the need is to give local language capabilities to utility softwares to cater the needs of diverse population of users [1]. A general student is not an expert in linguistics, statistics, or computer science but he has some expertise in his native language and prefers their native language for computer interaction. A majority of school-going children pursue their education in regional languages, among which Hindi language stands out to be most prominent. Schools are being provided with computer education facilities and internet connectivity so that vast educational resources already available and to be developed by schools themselves could be shared amongst them. Searching for

topics through table-of-contents or index pages can be tedious and impractical on account of large volume of information present in these domains. For instance, a user may be interested in looking for an answer to a question like "आधुनिक कृषि यंत्रों के उपयोग एवं उनसे होने वाली दुर्घटनाओं के विषय में बतायें". The complete answer may be in more than one passage that will be distributed throughout the corpus. So manual searching will be very cumbersome in this case. To effectively realize this foundational and organized setup of information resources, an effective information retrieval system is required.

As information needs are naturally represented as questions, a good solution is to have a QA system that provides direct answers rather than a ranked list of documents (as done by most search engines) to questions posed by the user by consulting its knowledge base. We have developed a QA System targeted at such students for access to information in E-learning documents (in Hindi) in a most user friendly way, that is by allowing them to pose question in Hindi and the system will return the intended answer by searching in context from the repository of Hindi documents. It is different from the keyword based search engines that match only patterns. Searching for a particular concept by keyword or phrase matching is insufficient because in many cases like for the question "गेहुँ का उत्पादन कैसे बढ़ाया जाएँ?" (How to increase the production of wheat?) Words like "उपज" or "फसल" may be there for "उत्पादन" or "वृद्धि" in place of "बढ़ाया" etc. Thus semantically related terms should be identified during search which is not done in normal search engine.

The system can supplement various Information and Communication Technology (ICT) applications aimed at benefiting general mass who are not English-literate. ICT has the potential of getting vast amounts of information to rural populations in a more timely, comprehensive and cost-effective manner. It can penetrate under-serviced areas and enhance education through distance learning. It can also facilitate development of relevant local content and faster delivery of information on technical assistance and basic human needs such as food, agriculture, health and water [2]. The internet can also enable the remotest village to access regular and reliable information from a global library (i.e., the web). Organized efforts are being made to empower the general mass through access to information and knowledge. In face of such a situation, a Hindi QA system becomes a necessary supplement for various ICT applications

2. Literature Review and Background

2.1 Related Work

Most of the research and development in multilingual question answering has been done in the development of cross–language information management systems. Research work has been done in Surprise Language Exercise (SLE) within the TIDES program where viability of Hindi-English Cross Lingual Question-Answering (CLQA) has been shown by [3] by developing a basic system. It accepts questions in English, finds candidate answers in Hindi newspapers, and translates the answer candidates into English along with the context surrounding each answer. However it is again aimed for English speaking users.

Another approach taken by some researchers at IIT Kanpur, India is that if source text is given in several languages, one can answer queries in some other language, without translating the sources into the language of the questioner [4]. Their approach is to convert queries and the documents into an intermediate representation, an inter–lingua called "the Universal Networking Language" (UNL). UNL is a

computer language suitable to represent contents. It represents the meaning of a text by building a (hyper) graph whose nodes are concepts and whose arcs are relations between concepts. They have attempted to demonstrate this kind of multilingual Question-Answering for source text in English and questions posed in Hindi and English. However the performance of the system, for any language, critically depends on the effectiveness of the enconversion (NL to Interlingua) and the de-conversion (Interlingua to NL). Venakata and Badodekar [5] discusses an implementation of a multilingual Question-to-Query conversion system in their attempt of integrating two independent systems called aAQUA and AgroExplorer. It converts English, Marathi and Hindi questions to syntactically correct and meaningful queries. AgroExplorer [6] is a meaning-based, multilingual search engine that considers the semantics of a query using UNL. aAQUA is an online multilingual, multimedia question and answer based community forum for disseminating information from and to the grassroots of the Indian community [7].

2.2 Background on Question Answering

There are many trends in question answering but in this paper, we only briefly introduce the systems most closely related to our system strategy. Specific research in the area of question answering has been prompted in the last few years in particular by the Question Answering track of the Text REtrieval Conference (TREC-QA) competitions [8]. Most of the QA systems that have been developed such as Mulder [9], AskJeeves [10], Webclopaedia [11] and LCC [12] treat the web as a collection of documents, that is, deal with open-domain question answering. The commercial search engine known as AskJeeves responds to natural-language questions but its recall is very limited because it uses its knowledge base (which is at least partially hand constructed) to answer questions and updates the knowledge base when asked a question which it has not encountered before.

Another QA system, MULDER is claimed to be the first general-purpose, fully-automated question-answering system available on the web. MULDER's architecture, relies on multiple search-engine queries, natural-language parsing, and a novel voting procedure to yield reliable answers (recall of same level as Google). However, the difficulty of Natural Language Processing (NLP) has limited their ability to give accurate answer to questions that are quite specific to a domain. In addition to the traditional difficulties associated with syntactic analysis, there remains many other problems to be solved, e.g., semantic interpretation, ambiguity resolution, discourse modelling, inference, common sense, etc.

2.3 Unicode Standard

Unicode Standard, by the California-based Unicode Consortium, is the universal character-encoding standard used for the representation of text for computer procession [13]. The World Wide Web Consortium (W3C) has recognized this fact and now expects all new RFCs (Request for Comments) use Unicode for text. Many other products and standards now require or allow use of Unicode; for example, XML, HTML, Microsoft JScript, Java, Perl, Microsoft C#, and Microsoft Visual Studio (C++ and Basic). We have developed our QA system using Microsoft Visual C++ as it provides good support (such as data types and string handling routines) to handle Unicode text.

The Unicode Standard, unlike ASCII, assigns each character a unique 16-bit value, which means that it can represent 65536 unique characters. Each of these 16-bit numbers is called a code value and, when referred to in text, is listed in

hexadecimal form following the prefix "U". For example the code value U+0041 is the hexadecimal number 0041(decimal 65). It represents the character "A" in the Unicode Standard. Each character is also assigned a unique name that specifies it and no other. For example, U+0041 is assigned the character name Latin character "A", U+0915 is assigned the character name Devanagari letter "KA" that is 'क'. The Unicode Standard 2.0 defines codes for characters used in the major languages written today. The coding starts at U+0000 with the standard ASCII characters, and continues with Greek, Cyrillic, Hebrew, Arabic, Devanagari, Bengali, Gurumukhi, Gujrati, Oriya, Tamil, Telegu, Kannada, Malayalam, Thai, Lao, Georgian, Tibetan, Japanese Kana, the complete set of modern Hangul, and a unified set of Chinese/Japanese/Korean (CJK) ideographs. The Unicode Standard does not define glyph images. The Standard defines how characters are interpreted, not how glyphs are rendered. The Unicode Standard specifies the order of characters used to create a composite character. The base character comes first, followed by one of more non-spacing marks. If text elements encoded with more than one non-spacing mark, the order in which the non-spacing marks are stored is not important if the marks do not interact typographically.

3. The Methodology and Architecture

The language constructs, query structure, common words, etc. are completely different in Hindi as compared to English. In the absence of a parser and a POS (Part of speech) tagger for Hindi finding the syntactic structure of a question is difficult. The task becomes more challenging as Hindi Wordnet is not released till our development period and the available morphological analyzer does not give satisfactory results. For converting the question to query, we rely on an approach of deleting words and use self constructed lexical database of synonyms to extend the query with semantically related terms. For improving the performance of search, we also implement a light weight stemmer for Hindi. By experimentation and analysis of question types, we developed case-based rules to classify the question. This classification helps in the later part of a*nswer selection* to put selectional restrictions to check which candidate answers satisfy the semantic constraints.

The architecture for the proposed QA System is shown in Figure 1. The user begins by configuring the system to the particular course domain by triggering the *Automatic Entity Generator* module which recognizes domain specific *entities* from that particular course documents. The question submitted by the user is classified in *Question Classification* to identify its case. The question is then parsed to separate out important Keywords by identifying the domain entities (based on the domain knowledge) and filtering out *Stop-words*. Subsequently Q*uery formulation* translates the question into a set of queries that is given as keyword input to the *Retrieval engine.* The engine returns top passages after weighting and ranking them on basis of locality. Finally *Answer selection* is done by further extensive passage analysis and presented to user.

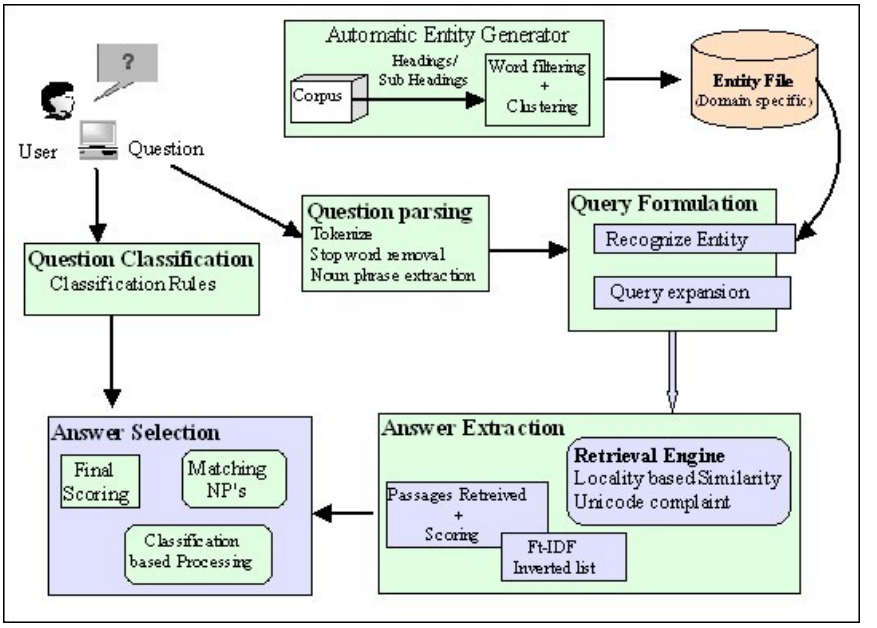

Figure 1. Architecture of the proposed QA system for Hindi

3.1 Automatic Entity Generator

This module tries to recognize the *entities* in a particular course (domain specific entity) to which the user wants to pose questions. This configures the system automatically to any type of course domain. The system administrator on the server providing distance learning (or the user who wants to search answer from documents present in his local system) gives the directory of files as input. The module then searches through the *Main heading* and *sub-headings* of the text files, recognizing them through their font size (larger than usual text size) and thus finds out the domain-specific entities. *Word filtering* is done to remove any elementary words. If no elementary words are found in the string then the whole string is also taken as an entity. The output is stored in the *Entity file* for subsequent use. This file contains domain specific entities.

3.2 Question Classification

Question classification is done by matching the patterns of interrogative words. Words like क्यों, क्योंकि, कहाँ etc are identified and questions are then put into the respective categories as described below.

- Questions which require reasoning and thus long explanations are identified by keywords like क्यों, क्या, वर्णन, कैसे.
- Questions that ask for numerical data like कितना.
- Questions that request for or indicated certain events like कब, जब.
- Questions which require persons as answer (किसने, किसको, कौन) or place (कहाँ, किधर).

- Questions that need answers from different passages like कौन-कौन, क्या- क्या, विभिन्न.
- Rest of the questions is put in this category and answers are identified on the basis of domain-specific entities or question-specific keywords as mentioned in the question.

This module basically helps in selection of appropriate answers from the results of retrieval engine, thus proving an important improvement over normal search engines.

3.3 Question Parsing

This module of parsing the question has been implemented at very elementary level in absence of a POS tagger for Hindi. In this module we find out the domain-specific and question-specific entities after removing the *Stop Words*. We also extract the longest phrase from the question which can then be used during the answer selection process to help find a more suitable answer to the question posed.

Stop Word Removal: In some languages such as English, functional words (e.g. "the", "a", "and", "that") are useless for indexing purposes. Similarly there are many words (as shown in Table 1) occurring in Hindi as well. These words occur in almost every document of the language, and therefore do not help in distinguishing between documents that are about different topics. For this reason, these functional words are removed and are not indexed. The process of removing these functional words is called stop words removal, and the functional words being removed are called stop words.

Table 1. Examples of removed words

में	से	है	होता
कि, का, के, की	भी	किया	गया
आता	जा	सकता	चाहिए
जाता	पर	तो	फिर
इधर	कर	कभी	इन

3.4 Query Formulation

The query formulation module transforms the question into a query which is eventually fed into the retrieval engine to extract answers. The system uses the entity file to recognize the domain specific entities in the question. During initialization, the system reads from default file (which can be set to a particular set of documents by the user) and constructs a hash table of these entities. Individual words in the question are compared from this table to identify the entities. These keywords are considered most important and are given the maximum weightage of 2. Stop words are given the weightage 0. Rest of the words in the question is given the weightage 1.

Query Expansion: Extending the query through query expansion enhances the search process by including semantically related terms and thus retrieves texts in which the query terms do not specifically appear [12]. For example words like उपयोग and प्रयोग, वक्त and समय, are used interchangeably. So inclusion of these synonyms

along with the keywords can help in better extraction of results. Due to non availability of Hindi Wordnet we had to construct a very limited lexical database of synonyms that served our experimentation purpose. For each word we have included two closest possible synonyms after consulting Hindi grammar books. With the Hindi Wordnet incorporated in the system, the semantic structure of the question can be tapped more effectively.

Only those query terms are expanded which do not occur as domain entities. Gaining from this knowledge, query evaluation is no longer restrained to query terms submitted by users but embodies synonymous or semantically related terms. However, caution is taken as these newly found terms are not as reliable as the initial terms obtained from users. Only closely related terms are taken that have direct relation with either the query term itself or with the words that are directly related to the query term. An appropriate weighting scheme with weight =0.5 allows a smooth integration of these related terms by reducing their influence over the query.

3.5 Answer Extraction

To extract passages from the collection of documents an information retrieval engine is needed which can analyze the keywords and passages in detail. The answers to a query are locations in the text where there is local similarity to the query, and the similarity is assessed by a mechanism that employs as one of its parameters the distance between keywords [14]. For this purpose it has been found that the locality-based similarity heuristic (in which every word location in each document is scored) provides good retrieval effectiveness because of the following features:

- The focus is on local context by considering top *n* ranked passages, instead of the top *n* documents.
- Each term has a certain scope, where its importance decreases with respect to the distance from that term
- Similarity is computed as the sum of weighted overlaps between terms. It is based on intuitive notion that the distance between terms is indicative of some semantics of the sentence.

Example Text Table

Document	Example Text
1	पशुओं में परजीवी कीटाणुओं से अनेक रोग फैलते हैं. प्रायः किसानों को इन रोगों के लक्षणों का पूरा ज्ञान न होने के कारण शुरू में रोगी पशु का पता नहीं लग पाता.
2	प्रायः परजीवी कीटाणुओं के प्रवेश करते ही पशुओं को दस्त लग जाते हैं.
3	रोग के लक्षण दिखाई देते ही किसान को पशु चिकित्सालय में पशु के गोबर की जांच करानी चाहिए.

A word Level Inverted Page List

Number	Words	Position (Document : Word)
1	परजीवी	(1:3), (2:2), (4:1), (4:7)
2	कीटाणुओं	(1:3), (2:3)
3	रोग	(1:7),(1:14),(1:26),(3:1)
4	पशु	(1:1),(1:27),(2:8) (3:9), (3:12)
….	….	………..

Figure 2: An example of construction of word-level inverted page list.

The first table shows some sample texts from three different documents

and second table shows the corresponding page list.

We used an available information retrieval engine SEFT (search for text) for English that uses this approach and modified it for searching in Hindi text. The implementation code has to be made Unicode complaint by changing the data types and string handling routines as specified in [15] to handle Hindi Unicode characters. The searching technique remains the same as described in [14]. The entire retrieval process is carried out using a word-level inverted index using all of the terms in the automatically generated query as shown in Figure 2. The text collection is considered to be a sequence of words rather than a collection of documents, and query term occurrences within the collection are presumed to exert an influence over a neighborhood of nearby words. Then, supposing that the influence from separate query terms is additive, the contribution of each occurrence of each query term is summed to arrive at a similarity score for any particular location in any document in the collection.

The contribution function C_t is then defined in terms of l, the location of the query term (as an integral word number); x, the word location at which we seek to calculate a contribution; h_t, the peak height assigned to the term, assumed to occur at the word position occupied by the term in question; and s_t, the one-sided spread of the term. The parameters that are used for scoring the passages are:

- N: total number of terms in the collection
- Term frequency (f_t): how often does term *t* appears
- $F_{q,t}$: Within query frequency of the term
- Inverse document frequency (*idf*): log (N/ f_t)
- Height (h_t): The height assigned to a term t is a monotonic function of the term's scarcity in the collection.
- $h_t = F_{q,t} * \log (N/ f_t)$
- d = |x – l| is the distance in words between the term in question and the location at which its influence is being evaluated. In each case the value of C_t *(x, l)* is defined to be zero when $|x - l| > s_t$

$$C_t (x , l) = h_t * \sqrt{(1 - (d / s_t)^2)}$$

The top ranked passages (window surrounding the location) is returned after scoring all the locations of the query term according to the weightage assigned to them. The implementation also handles *Stemming (*to match up a keyword with any

of its other grammatical forms*)* of word while searching the word and indexing them into the inverted page list.

Stemming of Hindi words:

Stemming is an operation that conflates morphologically similar terms into a single term without doing complete morphological analysis. Stemming is used in information retrieval systems to improve performance. Additionally, this operation reduces the number of terms in the information retrieval system, thus decreasing the size of the index files. In many languages, including Hindi and English, a word may exist in a number of morphological variants. For example, the English word compute also exists in other morphological variants such as 'computing', 'computed', 'computers' etc. Similarly the word लड़का can also exist in other morphological variant such as लड़कौ, लड़कें etc. While these morphological variants are different word forms, they represent the same concept. For indexing purposes it is desirable to combine these morphological variants into the same canonical form. This process is called word stemming and this canonical form is called root-word or base-word.

We implemented a lightweight stemmer for Hindi, which conflates terms by suffix removal based on the work given in [16]. The complete suffix list is shown in table 2. The Stemmer is implemented by simply removing from each word the longest possible suffix from this list.

Table 2. Suffix list

◌ा	◌ाएं	ता	◌ाने	◌ेगा
◌ि	◌ाओं	ती	◌ूंगा	◌ेगी
◌ी	◌ियां	ीं	◌ूंगी	◌ाएगा
◌ु	◌ियों	तीं	◌ाऊंगा	◌ाएगी
◌ू	◌ाइयां	ते	◌ाऊंगी	◌ाया
◌ै	◌ाइयों	◌ाता	◌ेंगे	◌ाए
◌ो	◌ाँ	◌ाती	◌ेंगी	◌ाई
◌ें	◌ियाँ	◌ातीं	◌ाएंगे	◌ाईं
◌ों	◌ाइयाँ	◌ाते	◌ाएंगी	◌िए
◌ां	ताएं	ना	◌ोगे	◌ाओ
◌ुआं	ताओं	नी	◌ोगी	◌ाइए
◌ुएं	नाएं	ने	◌ाओगे	कारा
◌ों	नाओं	◌ाना	◌ाओगी	◌ाकारा

3.6 Answer Selection and Presentation

The top ranked passages which are returned are answer candidates. These are further processed to select those answer passages that will be presented to the user. The system processes the passages according to the classification done in *question classification*. If the question was classified in the second or third category requiring any date or numerical expression then the system searches for these terms in the passages to match the answer type. For questions of other categories, we construct a list of related words that appear normally while answering specific type of questions. For example, the questions belonging to the fifth category having words like *विभिन्न* in them may have related words like *निम्न* or *मुख्यतः* in the answer passage. The answer passages are re-ranked using this information. Top three answers are presented along with the links to the specific location in relevant document as shown in Figure 3.

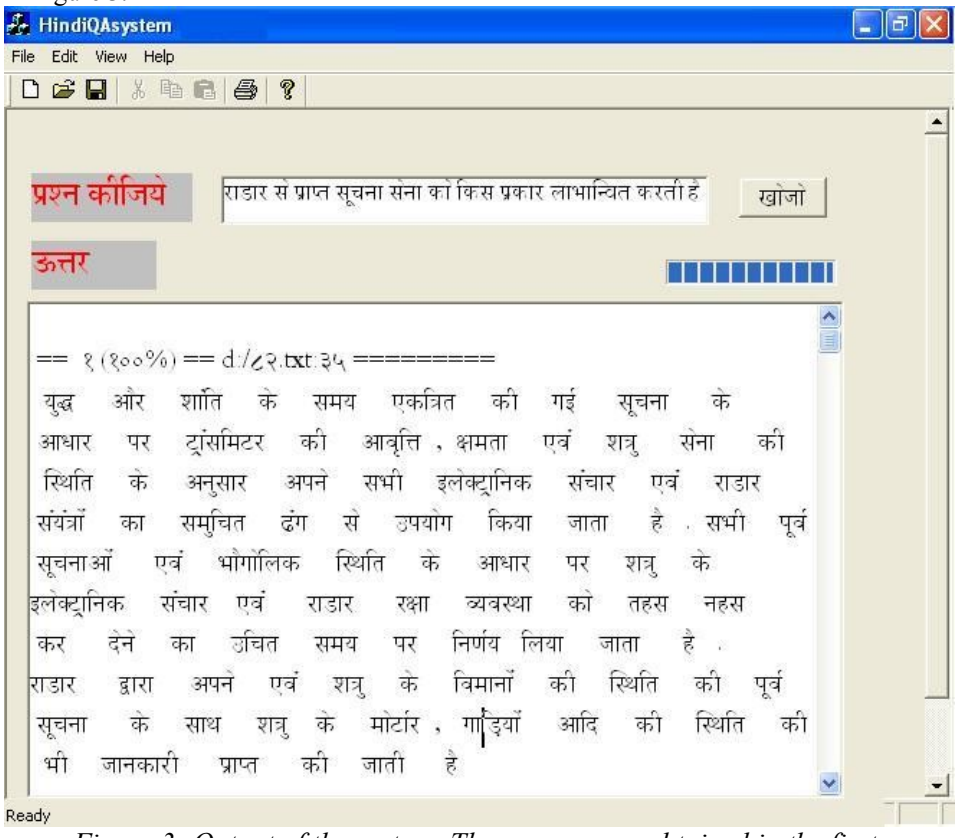

Figure 3: Output of the system: The answer was obtained in the first passage with full confidence (100%) giving also the information and link to the specific location in the relevant document

4. Experiments and Results

In order to perform experiment on the system, we took the corpus of Hindi Unicode documents available on the site of LTRC, IIIT Hyderabad, India. We selected two technical domains of agriculture and science and picked up those documents from the corpus that belong to these domains. For the purpose of testing, 30 questions were picked from a survey done on the students who read these files and 30 questions were formed by us to test the different aspects of implementation. The questions covered a wide range of topics on agriculture and science. They were of different types, complexity and difficulty. Questions were not only looking for facts rather they were explanation seeking. The result is shown in table 3.

Table 3: Experimental Results of System on our data set

Data set domain	Question type	# Questions (Q)	Answer (I)	Answer (II)	Answer (III)	Directs	Failed	Accuracy (I+II+III)/ Q (%)
Agriculture	Type 1	15	5	3	2	2	3	66.7
	Type 2	15	5	3	4	1	2	80.0
Science	Type 1	15	6	4	1	2	2	73.3
	Type 2	15	5	6	1	2	1	80.0
Overall		60	21	16	8	7	8	75.0

Type 1: Survey questions; Type 2: Our questions

The main goal of our experiments was to determine the efficiency of the system to locate the exact answers in either top 3 answer passages or direct the user to the relevant documents containing the answer (nearby to the retrieved passages). Three answers per query were extracted. The retrieval speed of QA system including information retrieval is fast enough to be negligible. The results of the questions were classified in six different categories as shown in the table 3. Questions that were answered in first, second and third passages are shown in the column *Answer I*, *Answer II* and *Answer III* respectively. Under the column *Directs,* those questions were included which were not answered directly in the retrieved passages but directed the user to the document containing the answer. Some questions which could not be answered by the system were included under the column *Failed*.

The system directly answered 75% of the questions in the three retrieved passages. For nearly 12% of the questions, the user was directed to the relevant documents. Finally the system failed to get the right answer in 13 % of the questions. Amongst these, nearly half of the questions were not within the purview of the material. Failure can be attributed to lack of clear boundary between classes of classification and inadequate use of syntactic information. The failure cases can be solved by extending the resources and classification rules.

5. Conclusions and Future Work

This paper presents a Hindi QA system that can be targeted to any restricted domain catering to Hindi documents in particular and uses NLP techniques to extract

answer passages. The implementation is done in Visual C++ and it tightly integrates Unicode compliant search engine with the various NLP modules build by us viz. light-weight parser, question classification module, query formulation module etc. Using the concept of entities the system is fully automated to work in any specific domain. The system is based on searching in context by using similarity heuristic and utilizes syntactic and partial semantic information. This achieves good accuracy in results.

The future work may include employing better methods for extracting semantic information to increase prediction accuracy. Hindi Wordnet and POS tagger can be used for better tapping of syntactic and semantic structure of the question. The current implementation utilizes only partial semantic information during answer extraction and selection. When successfully developed for courses written in Hindi language, it can then be generalized to cater to various fields like quizzing the websites for information as varied as the best seed available for planting cotton plants to the latest on government policies. It can go a long way in helping the needs of the villagers who are not English-literate. The system can be made multilingual by keeping the language independent modules of the architecture Unicode complaint and implementing the language specific modules like stemmer, question classification module and the graphical user interface for different languages like Marathi, Punjabi or any other language to be targeted.

References

[1] National IT Mission-Background Information http://www.mit.gov.in/E-rural/ (last accessed 30th April 2005)

[2] Bhatnagar, S.C., Subhash. *Information and Communication Technology in Rural Development: Case Studies from India.* World Bank Institute (WBI) Working Papers, WBI Publications, 2000, Pp. 1-13.

[3] Sekine, S., Grishman, R. Hindi-*English Cross-Lingual Question-Answering System.* ACM Transactions on Asian Language Information Processing, Vol. 2, No. 3, September 2003, Pages 181-192.

[4] Shukla, P., Mukerjee, A., Raina, A. *"Towards a Language Independent Encoding of Documents: A Novel Approach to Multilingual Question Answering."* In Proceedings of the 1st International Workshop on Natural Language Understanding and Cognitive Science, NLUCS 2004, Porto, Portugal, April 2004, pp. 116-125

[5] Venkata S.R.S.K, Badodekar, S., Bhattacharyya, P. *Question-to-Query Conversion in the Context of a Meaning-based, Multilingual Search Engine*. Symposium on Indian Morphology, Phonology and Language Engineering, IIT Kharagpur, February, 2005

[6] Sarvjeet Singh, M. Surve and P. Bhattacharyya. Agro-Explorer: A Meaning based Multilingual Search Engine. *International Conference on Digital Libraries (ICDL), New Delhi, India, Feb 2004*

[7] Ramamritham, K., Bahuman, A., Kumar, R., Chand, A., Duttagupta, S., Kumar G.V.R., Rao, C. *aAQUA - A Multilingual, Multimedia Forum for the community.* IEEE International Conference on Multimedia and Expo,2004.

[8] Voorhees, E.M. & Tice, D.M. *Implementing Question Answering Evaluation*. In Proceedings of LREC'2000 Workshop on Using Evaluation within HLT Programs: Results and Trends. 2000.

[9] Cody, C. T. K., Oren, E., & Daniel, S. W. (2001). *Scaling question answering to the Web.* Proceedings of the Tenth International Conference on World Wide Web, 150-161.

[10] Askjeeves: http://askjeeves.com/, 2000.

[11] Hovy E., Gerber L., & Hermjakob, U, Michael Junk and Chin-Yew Lin. *Question Answering in Webclopedia.* Ninth Text REtrieval Conference (TREC-9).Gaithersburg, MD. November 13-16, 2000

[12] Harabagiu, S., Moldovan, D. Pasca, M., Surdeanu, M. , Mihalcea, R., Girju,R.,Rus, M., Lacatusu,F., Morarescu, P. and Razvan Bunescu. *Answering Complex, List and Context Questions with LCC's Question-Answering Server*. Tenth Text REtrieval Conference (TREC-10).

[13] Unicode Home page *http://www.unicode.org*/

[14] Kretser, O.D. and Moffat, A. *Effective Document Presentation with a Locality-Based Similarity Heuristic.* Proc. 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, San Francisco, pp 113-20,1999.

[15] Unicode Enabled: Overview and Description. *http://www.unicode.org*, 2000.

[16] Ramanathan, A., Rao, D.D. *A Lightweight Stemmer for Hindi.* Proceedings of EACL, 2003.