Named Entity Based Answer Extraction form Hindi Text Corpus Using n-grams

Lokesh Kumar Sharma
Dept. of Computer Science and Engineering
Malaviya National Institute of Technology
Jaipur, India
2013rcp9007@mnit.ac.in

Namita Mittal
Dept. of Computer Science and Engineering
Malaviya National Institute of Technology
Jaipur, India
Nmittal.cse@mnit.ac.in

Abstract

Most existing systems, are constructed for the English language, such as state-of-art system Watson that win the Jeopardy challenge. While working with Indian languages (i.e. Hindi), a richer morphology, greater syntactic variability, and less number of standardized rules availability in the language are just some issues that complicate the construction of systems. It is also considered a resource-poor language since proper gazetteer lists and tagged corpora is not available for it. In this paper, Named Entity (NE) based n-gram approach is used for processing questions written in Hindi language and extract the answer from Hindi documents. Combination of classical information retrieval term weighing model with a linguistic approach mainly based on syntactic analysis is used. We use a corpus of 420 questions and 300 documents containing around 20,000 words as a back-end for closed-domain (World History) Question Answering. A Named Entity Recognizer is employed to identify answer candidates which are then filtered according their usage. Results obtained using this technique outperforms the previously used techniques (e.g. Semantic Based Query Logic).

1 Introduction

With the advancement in technology, Question Answering has become a major area of research. Question Answering systems enable the user to ask questions in natural language instead of a query and retrieve one or many valid and accurate answers in natural language. The explosion of information on Internet, Natural language QA is recognized as a capability with great potential (Hirschman and Gaizauskas, 2001). Information retrieval systems allow us to locate full documents or best matching passages that might contain the pertinent information, but most of them leave it to the user to extract the useful information from a ranked list. Therefore, professionals from various areas are beginning to recognize the usefulness of other types of systems, such as QA systems, for quickly and effectively finding specialist information. The QA technology takes both IR and IE a step further, and provides specific and brief answers to the user's questions formulated naturally. Hindi holds 5th position among top 100 spoken languages in the world, with no. of speakers being close to 200 million (Shachi et al., 2001) but comparing Indian languages with other languages, word segmentation is a key problem in Indian question answering. As per our knowledge not much work has been done in Hindi as compared to various other languages like English (Ittycheriah et al., 2008), Chinese etc. This motivates for developing a Hindi question answering system (Vishal and Jaspreet, 2013). Our dataset consists of 420 questions and 300 documents containing around 20,000 words chosen from a specific domain (World History). Our model involves three general phases which are as follows. The first phase, Question Processing, involves analyzing and classifying the questions into different categories. This classification later helps in Answer type Detection. Further, in this module, a query is formulated which is passed on to the next phase for searching the relevant documents which might contain the answer. In the second phase, Information Retrieval, we have applied an algorithm called Term Frequency-Inverse-Document-Frequency

D S Sharma, R Sangal and J D Pawar. Proc. of the 11th Intl. Conference on Natural Language Processing, pages 362–367,
Goa, India. December 2014. ©2014 NLP Association of India (NLPAI)

(TF-IDF) (Ramos, 2003), which uses dot product and cosine similarity rule to find the probability of a given text in a given set of documents. This gives us the list of relevant documents. The next phase, Answer Extraction, uses bigram forming approach (Wang et al., 2005) to retrieve the answer from a given document. In this we have also used a prebuilt Hindi named entity recognition model which categorizes the given text into different categories.

2 Related Work

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 (Satoshi and Ralph, 2003). Recently IBM Watson defeated two human winners and win the Jeopardy game show. Watson uses very complex algorithm to read any given clue. At the first stage in question analysis Watson does parsing and semantic analysis using a deep Slot Grammar parser, a named entity recognizer, a co-reference resolution component, and a relation extraction component (Lilly et. al 2012). Our work uses similar approach by using named entity taggers and parsing. Research work has been done in Surprise Language Exercise (SLE) within the TIDES program where viability of a cross lingual question answering (CLQA) (Shachi et al., 2001) has been shown by developing a basic system. It presents a model that answers English questions by finding answers in Hindi newswire documents and further translates the answer candidates into English along with the context surrounding each answer (Satoshi and Ralph, 2003). Another approach taken by some researchers (Praveen et al., 2003) presents a Hindi QA system based on a Hindi search engine that works on locality-based similarity heuristics to retrieve relevant passages from the corpus over agriculture and science domain. Some researchers (Sahu et al., 2012) discusses an implementation of a Hindi question answering system "PRASHNOTTAR". It presents four classes of questions namely: "when", "where", "what time" and "how many" and their static dataset includes 15 questions of each type which gives an accuracy of 68%. In addition to the traditional difficulties with syntactic analysis, there remains many other problems to be solved, e.g., semantic interpretation, ambiguity resolution, discourse modeling, inference, common sense etc.

3 Proposed Approach

Question Processing is the first phase of our proposed question answering model in which we analyze the question and create a proper IR query which is further used to retrieve some relevant documents which may contain the answer of the question. Another task is question classification to classify a question by the type of answer it requires. The former task is called Question Classification and the latter one is known as Query Formulation. Both these aspects are equally important for Question Processing.

3.1. Question Classification

The goal of Question Classification is to accurately assign labels to questions based on expected answer type. Hence, we detect the category of a given question.

Question Phrase	Answer Type (AT)
क्या	AT:Desc, Single type cannot be decided
कब	Date
कहाँ	Location
कितनी कितना कितने	Number
कौनसा कौनसी	Answer type depends on next following word
किसका किसकी कौन किसे किसने	Person
क्यों	AT:Desc, Single type cannot be decided
कैसे	AT:Method, Single type cannot be decided
किस	Answer type depends on next following word

Table 1. Possible Answer Type Based on Question Phrase

In English there are 6 main categories namely LOCATION, PERSON, NUMERIC, ENTITY, ABBREVIATION and DESCRIPTION and but for

Hindi we have taken only 4 categories for our categorization process includes PERSON, COUNT, DATE and LOCATION. We applied proposed algorithm over the following answer types highlighted in table 1. The output file contains the previously mentioned category of question to which it belongs followed by the question itself and thus mapping from questions to answer types is done here. After categorization of the question, we store it in a file, so that it can be used later for answer extraction. Here is an example. Suppose we have the following question, लोक अदालत की शुरुआत राजस्थान में सबसे पहले कहां हुई ? Then the output file will contain: LOCATION: लोक अदालत की शुरुआत राजस्थान में सबसे पहले कहां हुई।

3.2. Query Formation

Query Formation is a technique to make the question format such that it can be passed on to a system which takes the input in the form of a query and searches out the relevant documents i.e. the documents which have the maximum probability of containing the answer. For this purpose, we have formulated a query by extracting the main or focus words (Haung, 2008) of the question by removing the stop words occurring in the question. For this, we have used a file containing a prebuilt list of stop words. Examples of some stop words are: (के, का, हुई, है, पर, इस, होता, , बनी, नहीं, तो, ही, या, एवं, दिया, हो, इसका, था, द्वारा, हुआ, तक, साथ, करना, कुछ, सकते, किसी, हुई) After stop words removal, the text looks like this: लोक अदालत शुरुआत राजस्थान पहले After removal of these less important words from the question, the resultant output can be used as a query for the information retrieval system which involves the next part of the model.

3.3. Relevant Information Extraction

The task of Information Retrieval phase is to query the IR Engine, find relevant documents and return candidate passages that are likely to contain the answer. In our model, our dataset is scattered over various documents, each containing question related text along with its answer. Then we performed a search within these documents in order to find out such documents which may contain the answer. And for this purpose, we have applied an algorithm called TF-IDF; it gives as output the list of various documents which may contain any of the given words from the query. The term frequency (TF) for a given term t_i within a particular document d_j is defined as the number of occurrences of that term in the d_j^{th} document, which is equal to $n_{i,j}$: the number of occurrences of the termt$_i$ in the document d_j.

$$TF_{i,j} = n_{i,j}$$

IDF(t_i) = log$_e$(Total number of documents / Number of documents with term t in it).

$$IDF_i = \frac{log\ |D|}{|\{d: ti \in d\}|}$$

With |D|: total number of documents in the collection and $|\{d : t_i \in d\}|$: number of documents where the term t_i appears. To avoid divide-by-zero, we can use $1 + |\{d : t_i \in d\}|$. For a given corpus D, then the TF-IDF is then defined as:

$$(TF\text{-}IDF\ )_{i,j} = TF_{\ i,j} \times IDF_{i.}$$

The input of TF-IDF is the file which contains focus words of the question i.e. the output of query processing. When TF-IDF algorithm was applied on this file, it gave as output the relevant documents i.e. documents having maximum probability of containing the answer. TF-IDF numbers imply a strong relationship with the document they appear in, suggesting that if that word were to appear in a query, the document could be of interest to the user (Ramos, 2003). For our given example, the given method extracted the following:

औधोगिक विवादों के त्वरित निपटारे के किए जयपुर स्थित राजस्थान उच्च-न्ययालय में 20 जुलाई को मेगा लोक अदालत का आयोजन किया जाएगा । लोक अदालत की शुरुआत राजस्थान में सबसे पहले कोटा में हुई । **</146.txt>**

146 is the document number from which it extracts the passage. Then we take the mentioned documents and retrieve all its content in a separate file which is further used to find answers in the answer extraction phase.

3.4. Answer Extraction

The final task of a QA system is to process the relevant Passages (which we get after Information Retrieval phase) and extract a segment of word(s) that is likely to be the answer of the question. Question classification comes handy here. There are various techniques for answer extraction. We have used the following steps to extract answer.

Step 1: Take the file containing the text and remove all of its stop words.

Step 2: Take the file which contains the question and form its bigrams i.e. form words taking twice a time and stored it in a file.

Step 3: Then take the output file and form the bigram of the text it contains and match it with the file which contains the question's bigrams.

Step 4: Save the number of bigrams matched for each line to the question's bigrams.

Step 5: Output the line which contains the maximum number of bigrams matched.

We have the following output after removing stop words from the passage:

औधोगिक विवादों त्वरित निपटारे जयपुर स्थित राजस्थान उच्च-न्ययालय 20 जुलाई मेगा लोक अदालत आयोजन जाएगा लोक अदालत शुरुआत राजस्थान पहले कोटा

After storing this output file as a target document. The questions are stored in a separate file of their bigrams i.e. taking two words together (Wang and McCallum, 2005). Storing the outcome in a file called QBigram-feature file. This gave us the following output,

Q-Bi-gram$_{\textbf{(feature)}}$ = {(लोक अदालत)$_1$, (अदालत शुरुआत)$_2$, (शुरुआत राजस्थान)$_3$, (राजस्थान पहले)$_4$, (पहले हुई)$_5$}

The given passage will have following bigrams,

P-Bi-gram$_{\textbf{(feature)}}$ = **{(**लोक अदालत)$_1$, (अदालत शुरुआत)$_2$, (शुरुआत राजस्थान)$_3$, (राजस्थान पहले)$_4$, (पहले कोटा)$_5$, (कोटा हुई)$_6$}

Now these bigrams will be matched with the question's bigram as per our designed algorithm. The concept in this is, the line which contains the maximum number of two words same at a time will have maximum probability of containing the answer. So when we do this we will get following line as output:

लोक अदालत की शुरुआत राजस्थान में सबसे पहले कोटा में हुई । </146.txt>

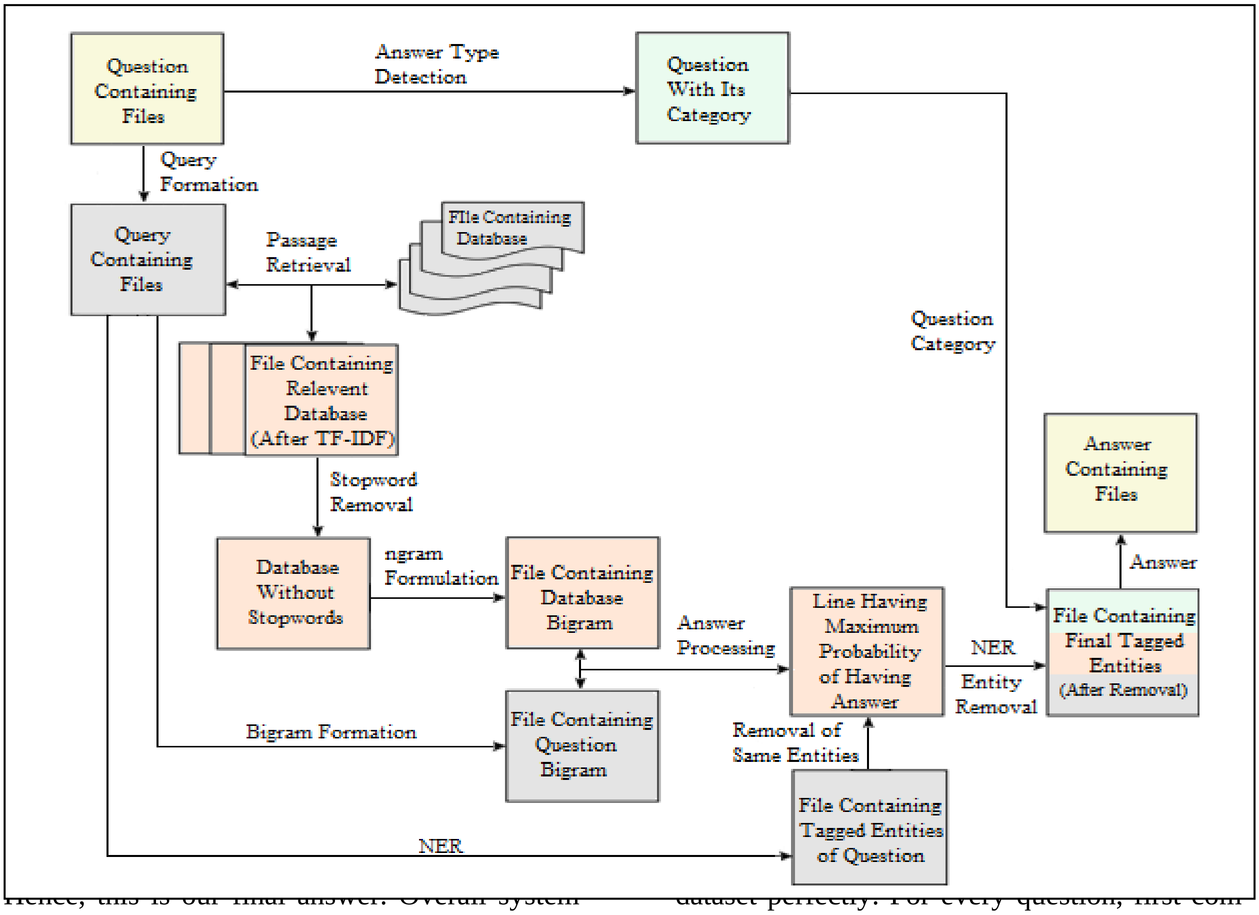
Now we pass the question containing file to the prebuilt Hindi Named Entity Recognition (NER) System (Maksim and Andrey, 2012) which will tag the given text into the aforementioned 5 categories. The NER gives output as following:

लोक	o	both
अदालत	o	both
शरुआत	o	both
राजस्थान	LOCATION	GAZETTEER
पहले	o	both

As we know the possible type of answer from the question classification method which we have applied earlier, we can remove those named entities which are present in both answer and the question, as they will not be the required answer. And hence, the remaining tagged entity will be our required answer. After removing the named entities which are tagged in the question, following words are left in the text: लोक अदालत शुरुआत पहले कोटा Now running the NER on the output line again, getting the tagged output:

लोक	o	both
अदालत	o	both
शरुआत	o	both
पहले	o	both
कोटा	LOCATION	GAZETTEER

Through this output, we extract the entities which matches the Answer Type which we have detected earlier i.e. Answer Type Detection (Roberts and Hickl, 2008) is done on the output. For example in our case here the Answer Type is LOCATION, so we extract the entity which is tagged as location which is <**कोटा**>.


Question Containing Files
Answer Type Detection
Question With Its Category
Query Formation
Query Containing Files
Passage Retrieval
FIle Containing Database
Question Category
File Containing Relevent Database (After TF-IDF)
Stopword Removal
Answer Containing Files
Database Without Stopwords
ngram Formulation
File Containing Database Bigram
Answer
Answer Processing
Line Having Maximum Probability of Having Answer
NER Entity Removal
File Containing Final Tagged Entities (After Removal)
Bigram Formation
File Containing Question Bigram
Removal of Same Entities
File Containing Tagged Entities of Question
NER

Hence, this is our final answer. Overall system architecture is shown in figure 1.

4. Experimental Setup and Analysis

To evaluate the effectiveness of the proposed methods for answer extraction from Hindi corpus, 300 standard documents datasets is used. The accuracy for the questions of category ‘कब’, ‘कहाँ’, ‘कितनी, कितना, कितने’, and ‘किसका, किसकी’ is satisfactory in proposed approach shown in table 2. The accuracy of question type ‘किस समय’ is not considered by the proposed approach because the answer type of this question has been not considered. The accuracy of the question type ‘कब’, ‘कहाँ’, and ‘किसका किसकी’ is highly accurate. Some question has low syntactic information to reach the answer, and it is difficult for the system to answer. For such a questions it may have multiple documents and multiple matches in these documents, an algorithm may not extract every answer in the dataset perfectly. For every question, first compute its precision (P) and its recall (R) by taking the dataset as gold standard answers as the relevant answer and the predicted answer at the retrieved set. Now, taking an average of P and R over all Topics. Now, calculating macro F1 using the harmonic mean of the average P and R,

$$\boldsymbol{macro\ F1} = \frac{\boldsymbol{2PR}}{(\boldsymbol{P}+\boldsymbol{R})}$$

Where,

$$P = \frac{correct}{retrieved} \text{ and } R = \frac{correct}{retrieved} .$$

Accuracy (F1-measure) is calculated which outperforms existing Semantic Based Query Logic approach comparison results are highlighted in the Table 2.

Type of Question (Total 420 Question)	Accuracy (macro F1)	
	(Semantic Based Query Logic)	**(Proposed Approach- NE Based n-gram)**

कब	66.66%	**74.33%**
कहाँ	53.00%	**86.66%**
कितना कितने कितनी	73.33%	**72.50%**
किसका किसकी	-	**82.75%**
Total	64.33%	**79.06%**

Table 2. Accuracy of the proposed approach

The question set of 420 questions[1] and supported answer documents used in this work are manually collected from web. The documents have answer for every question still it is not easy to extract correct answers for all questions.

5. Conclusion and Future Work

In this paper, Question answering for Hindi language has been experimented on 420 natural language questions. Results outperforms the previously used semantic based logic query approach. Using this approach, we achieved state-of-art results for most of the question types namely Person, Location, Date and Count. But as this approach is syntactic, so using this approach we able to get answers for factoid questions. Text where usage of synonyms or hyponyms of words is seen, accurate answers could not be extracted. Such issues can be dealt by introduction of the semantic approach. Results can be improved by adding features like entailment, co-reference etc in the answer extraction phase. Improving the accuracy of Hindi NER will also help in improving the accuracy of the system. Also, as our model is domain based, one can extend its domain by using a searching algorithm over the Wikipedia or other online resources.

References

Hirschman L., Gaizauskas R., *“Natural language question answering: the view from here”*, Natural Language Engineering, v.7 n.4, p.275-300, December, 2001.

Ittycheriah et al., *“IBM’s Statistical Question Answering System”*, In Proceedings of the Ninth Text Retrieval Conference (TREC-9), 2000. Roberts, K., & Hickl, A, “Scaling Answer Type Detection to Large Hierarchies”, In Proceedings of LREC, May 2008.

Lally A., Prager J. M., McCord M. C., Boguraev B. K., S. Patwardhan, Fan J., Fodor P., and Chu-Carroll J., *“Question analysis: How Watson reads a clue”*, IBM J. Res. Dev., vol. 56, no. 3/4, Paper 2, pp. 2:1–2:14, May/Jul. 2012.

Maksim Tkachenko, Andrey Simanovsky, *“Named Entity Recognition: Exploring Features”*. In Proceedings of KONVENS 2012, Vienna, September 20, 2012.

Praveen Kumar, Shrikant Kashyap and Ankush Mittal, *"A Query Answering System for E-Learning Hindi Documents"*, South Asian Language Review Vol. XIII, Nos. 1&2, January-June,2003.

Ramos J., *“Using TF-IDF to determine word relevance in document queries”*. In Proceedings of the First Instructional Conference on Machine Learning, December 2003.

Roberts K. and Hickl A, *“Scaling Answer Type Detection to Large Hierarchies”*, In Proceedings of LREC, May 2008.

Sahu S., Vasnik N., and Roy D., *“Prashnottar: A Hindi Question Answering System”*, International Journal of Computer Science and Technology, Vol 4, pp. 149-158, 2012.

Satoshi Sekine and Ralph Grishman, *“Hindi-English cross-lingual question-answering system*”, ACM Transactions on Asian Language Information Processing (TALIP), v.2 n.3, p.181-192, September 2003.

Shachi Dave, Pushpak Bhattacharya & Dietrich Klakowya, *“Knowledge Extraction from Hindi Text”*, Journal of Institution of Electronic and telecommunication engineers, 18(4), 2001.

Vishal G. and Jaspreet K., *“Comparative Analysis of Question Answering System in Indian Languages*”, International Journal of Advanced Research in Computer Science and Software Engineering” 3(7) pp.584-592, July 2013.

Wang X. and McCallum A., *“A note on topical n-grams”*, Massachusetts University Amherst Dept of Computer Science, 2005.

[1] https://code.google.com/p/hindiqset/