Design and Implementation of an Efficient Search Engine with Bangla Interface using NLP

Conference Paper · June 2008

CITATIONS
0

READS
455

2 authors, including:

Mahmudul Hasan
Saitama University
33 PUBLICATIONS **71** CITATIONS

SEE PROFILE

International Conference on Electronics, Computer and Communication (ICECC 2008)
University of Rajshahi, Bangladesh

Design and Implementation of an Efficient Search Engine with Bangla Interface using Natural Language Processing (NLP)

Sujan Kumar Roy, K.M Ibrahim Khalilullah, Md. Ikbal Aziz Khan, Mahmudul Hasan
Dept. of Computer Science & Engineering, University of Rajshahi, Rajshahi, Bangladesh
sujan_5401@yahoo.com, cse_5403@yahoo.com, ikbal_aziz_khan@yahoo.com, cse_raju@yahoo.com

ABSTRACT: *The design and implementation of an efficient bangla interfaced search engine using Natural Language Processing (NLP) has been presented in this paper. The search engine provides dynamic bangla interface so that general people of bangla speaking can use it more frequently and easily. A user of the search engine can input bangla sentence without any help from the third party. Some new techniques of Natural Language Processing (NLP) are used to enhance searching capability. A database is designed so that the search engine can use this database to return more appropriate result.*

KEYWORDS: NLP, Search Engine, Dynamic Bangla Interface, Bangla Interpreter, Independent Key- map, Unicode, Lexicon, Top-Down parser.

1. INTRODUCTION

Search engines are forced to issue millions of successive queries resulting in unnecessary search engine load and in slow applications with limited scalability.

Natural language processing applications perform computations over large corpora. With increasing frequency, NLP applications [4] use the web as their corpus and rely on queries to commercial search engines to support these computations. But search engines are designed and optimized to answer people's queries, not as building blocks for NLP applications.

2. ARCHITECTURE OF THE SEARCH ENGINE

The architecture of the search engine is as follows:

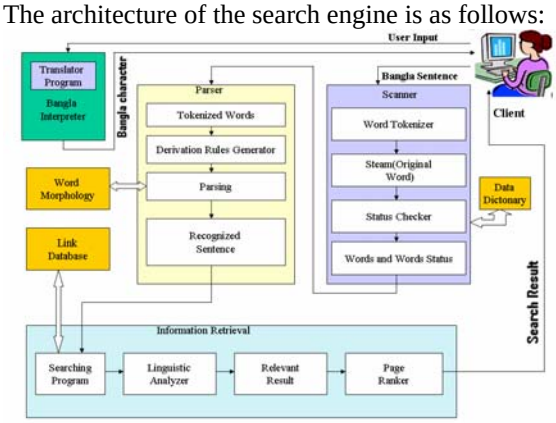

Fig. 1: Block diagram of the search engine.

Different phases of the system have been described below.

3. DYNAMIC BANGLA INTERFACE

The interface provided by the engine is friendly and dynamic. The contents (e.g. texts images, animations and so on) of the interface dynamically change with respect to time. All changes provides different graphical user interface, which is more locative and attractive. So, the user always get enjoy about it when access the search engine in different times. The layout of dynamic bangla interface is as follows:

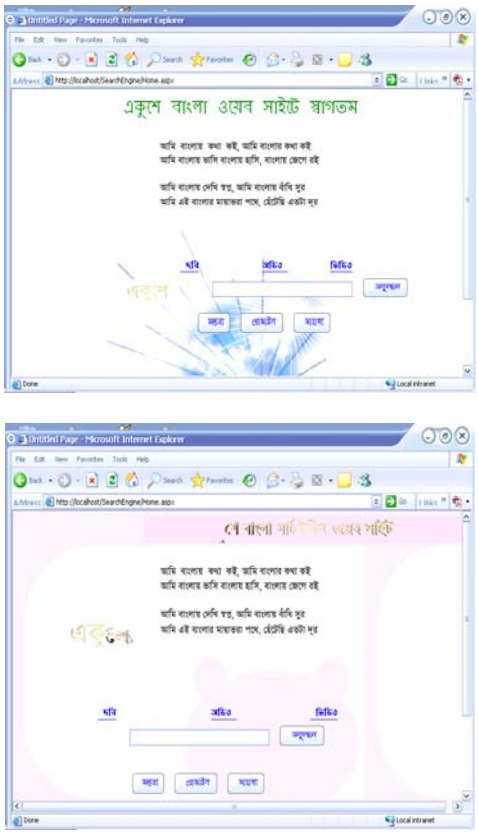

Fig. 2: Dynamic bangla interface.

4. BANGLA INTERPRETER

When users press a key from keyboard, the bangla translator program [3] receives that character's key code and converts it into a bangla character. The program use Unicode interfacing [3] for obtaining bangla character code corresponding to an English character typed by user. The bangla typing capability is handled by own independent key-map.

The key-map is as follows:

Fig. 3: Independent key-map for general bangla character.

Fig. 4: Independent key-map for Kars and Folas.

Some examples of bangla typing are given below:
মস্ত=mos+To
অর্ক=aor+ko
মুখোশ=mukosh

5. NLP ENGINE

The NLP engine [6] [8] receives bangla sentence as input. It processes the query and generates a recognizable sentence. The query processing steps has been described in the next section.

5.1 Query Processing

There are several phases of query processing. At first, we have to consider the suffixes that are combined with the words of the sentence. As for example some well known suffixes are**:** টা, টি, রা, রে,কে,জন,গন,গুলি,গুলো,দের,র,এর,খানা,খানি,গুলো,এরা,য়,তে,ই,ন,ছ,ল,ছি,বে,তাম etc. Now we have to discard the suffixes from the words and find out the root words by Stemming algorithm [15]. The stemming algorithm used to remove suffix is as follows:

1. Receive the input sentence and separates the words of that sentence to store individually.
2. Now match the stored words with the words in the lexicon.
3. If a match is found then this stored word is meaningful word and marked as a root word and store separately.
4. If no match is found then for each word perform the following test:

   4(a). Compare the sub strings (টা,টি,রা,রে,কে,জন,গন,গুলি,গুলো,দের,র,এর,খানা,খানি,গুলো,এরা,য়,তে,ই,ন,ছ,ল,ছি,বে,তাম) with the words.

   4(b). If a match is found then discards the substring from the word and store the remaining part of the words.

   4(c). If no match is found then store the word without any change.

5. End.

The following table represents some root word after discarding suffixes.

Word	*Stem*	*Suffix*
করছিলাম	কর	ছিলাম
নড়ছেন	নড়	ছেন
ঠেলাচ্ছিলাম	ঠেলা	চ্ছিলাম
জমাচ্ছ	জমা	চ্ছ
মারলাম	মার	লাম
পড়াতি	পড়া	তি
কচলাচ্ছিলে	কচলা	চ্ছিলে
আটকাব	আটকা	ব
হাতড়াচ্ছি	হাতড়া	চ্ছি
পাকড়ালেন	পাকড়া	লেন

Fig. 5: The result of our stemming algorithm.

5.2 Lexicon

A lexicon is a dictionary of words, where each word contains some syntactic and semantic information. For our search engine, the lexicon contains only the root words and their status (like noun, pronoun and verb).

5.3 Translate the Query in Recognizable Form

The output of the lexical analyzer is treated as the root words. Now with this root word we generate a recognizable search topic. For this purpose rule based Top-Down parsing technique [9] is used.

5.3.1 Top-Down Parsing Technique

A top down parser begins by hypothesizing a sentence and successively predicting lower level constituents until individual terminal symbols are

written. The parsing algorithm [2] [4] that we use to make a sentence in a recognizable form is as follows:

1. From input sentence, separate all tokens and store those individually.
2. Call the sentence rule with the input sentence, which subsequently calls phrase rule with first token of the sentence. The phrase rule then checks the word's status (e.g. noun, pro. Noun etc), if the word satisfy the requirements of the phrase rule then the parser rule gets the next token from the sentence and check it as earlier, this process continues until a phrase is detected. If any word fails to satisfy the requirements of a rule, the parser automatically back track to the next phrase rule not examined and the above process repeats.
3. Repeat step 2 for all the words of the sentence.
4. If all the phrases in the sentence are identified then class agreement between noun phrase and verb phrase is checked and form the recognized sentence.

The parser has been designed by rule based Context-Free Grammar (CFG).

For example, consider the following grammar:

S→ NP VP
NP→ N | P
VP→ V OBJ
OBJ→ NP
N→ মোবাইল। প্রোগ্রাম। জেটুএমই
V→ সাহায্য

The top-down parsing for the sentence "জেটুএমইর সাহায্যে মোবাইল প্রোগ্রাম" is given by,

S→ NP VP
S→ N VP
S→ জেটুএমই VP
S→ জেটুএমই V OBJ.
S→ জেটুএমই সাহায্য OBJ.
S→ জেটুএমই সাহায্য NP
S→ জেটুএমই সাহায্য N
S→ জেটুএমই সাহায্য মোবাইল N
S→ জেটুএমই সাহায্য মোবাইল প্রোগ্রাম

The top-down parse tree for the above derivation has shown in the following figure.

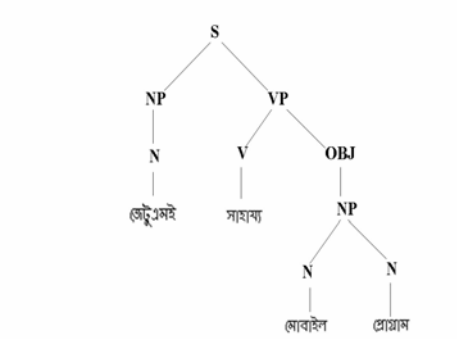

Fig. 6: Parse tree for **"**জেটুএমইর সাহায্যে মোবাইল প্রোগ্রাম **".**

6. SEARCHING INFORMATION FOR USER QUERY

6.1 The Link Database

The link database performs two functions. First it manages the task of assigning a globally unique ID for every link that the crawler has identified. The second functionality is to store various static properties of the URLs that Infocious is aware of. Such information includes the number of incoming links to a Web page, the number of outgoing links, a content signature, a concept list signature, and the quality of the text described earlier. This information is used during the ranking of results, as well as for rescheduling crawls.

6.2 Information Retrieval Technique

Given a user query q and a document d_j in the collection, the probabilistic model [14] tries to estimate the probability that the user will find the document d_j interesting (i.e., relevant). The model assumes that this probability of relevance depends on the query and the document representations. Further, the model assumes that there is a subset of all documents which the user prefers as the answer set for the query q. Such an ideal answer set is labeled R and should maximize the overall probability of relevance to the user. Documents in the set are predicted to be relevant to the query. Documents not in this set are predicted to be non-relevant. This assumption is quick troublesome because it does not state explicitly how to compute the probabilities of relevance. In fact, not even the sample space, which is to be used for defining such probabilities, is given.

Given a query q, the probabilistic model [13] [14] assigns to each d_j, as a measure of its similarity to the query, the ratio P (d_j relevant to q)/P (d_j non-relevant to q) which computes the odds of the document d_j being relevant to the query q.

For the probabilistic model, the index term weight variables are all binary i.e., $w_{i,j} \in \{0, 1\}$, $w_{i,q} \in \{0, 1\}$. A query q is a subset of index terms. Let R be

the set of documents known (or initially guessed) to be relevant. Let $\bar{R}$ be the complement of R. Let P (R/ $\bar{dj}$) be the probability that the document dj is relevant to the query q and P ($\bar{R}$ / $\bar{dj}$) be the probability that d_j is non-relevant to q. The similarity sim ($d_{j,q}$) of the document d_j to the query q is defined as the ratio:

sim (d_j q) = P (R/ $\bar{dj}$)/P ($\bar{R}$ / $\bar{dj}$).

Using Bayes' rule [14]:

$$sim(dj,q) = \frac{P(\bar{dj}/R) \times P(R)}{P(\bar{dj}/\bar{R}) \times P(\bar{R})} \quad .$$

$P(\bar{dj}/R)$ stands for the probability of randomly selecting the document d_j from the set R of relevant documents. Further, P(R) stands for the probability that a document randomly selected from the entire collection is relevant.

6.3 Page Rank Calculation

Academic citation literature has been applied to the web, largely by counting citations or back links to a given page. This gives some approximation of a page's importance or quality. Page Rank [11] [13] extends this idea by not counting links from all pages equally, and by normalizing by the number of links on a page. Page Rank is defined as follows: We assume page A has pages T1...Tn, which point to it (i.e., are citations). The parameter d is a damping factor, which can be set between 0 and 1. We usually set d to 0.85. Also C (A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

PR (A) = (1-d) + d (PR (T1)/C (T1) + ... + PR (Tn)/C (Tn))

Note that the Page Ranks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.

PageRank or *PR (A)* can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web. Also, a PageRank for 26 million web pages can be computed in a few hours on a medium size workstation.

7. COMPARISON OF THE SEARCH ENGINE WITH OTHER SEARCH ENGINES

Due to use of NLP, the user queries are transformed to the exact computer recognizable form for searching. For information retrieval, in this search engine, probabilistic model has been used, which retrieves appropriate search results. The rank of a page has been calculated by a probability theorem, so it displays more desired results with respect to the user query in the first page, less desired results in the second page and son on.

8. CONCLUSION

Natural Language Processing (NLP) processes search queries in a way that the search engine returns only desired results than other search engines, which provide some undesired results with desired results. The search engine will take a large contribution in search engine application areas especially for bangla speaking people all over the world.

References

[1] E. Spertus and L. A. Stein. Squeal, "A Structured Query Language for the Web". *In Proceedings of the 9th International World Wide Web Conference (WWW9)*, pages 95-103, 2000.

[2] Brill, E. and R.J. Mooney, "An overview of empirical natural language processing", *AI Magazine Winter (1997)*: 13-24.

[3] D. Bahle, H. E. Williams, and J. Zobel. Optimized, "Phrase Querying and Browsing in Text Databases". *In M. Oudshoorn, editor, Proceedings of the Australasian Computer Science Conference*, pages11 -19, Gold Coast, Australia, Jan. 2001.

[4] Kreymer, Oleg, "An evaluation of help mechanisms in natural language information retrieval systems", *Online Information Review 26 (2002):* 30-39.

[5] Microsoft Developer Network (MSDN) 2005.

[6] Matt J. Crouch, "ASP.NET and VB.NET Web Programming." *First Indian Reprint, 2002.*

[7] Dino Esposito, "Programming Microsoft ASP.NET 2.0, Microsoft 2005 Edition"

[8] Danny Goodman, JavaScript™ Bible, 4th Edition.

[9] C. D. Manning and H. Schutze, "Foundations of Statistical Natural Language Processing". *MIT Press*, 1999.

[10] Kim, K. S. and Allen, B, "Cognitive and task influences on Web searching. "*JASIST*, 52(2): 109-119.

[11] Lawrence, S. (2000), "Context in Web search.", *IEEE Data Engineering Bulletin*, 23(3): 25-32.

[12] Muramatsu, J. and Pratt, W, "Transparent queries: Investigating users' mental models of search engines". *In Proc. of SIGIR* '01, 217-224.

[13] Spink, A., Wolfram, D., Jansen, B. J. and Saracevic, T.(2001), "Searching the Web: The public and their queries". *Journal of the American Society for Information Science*, 52(3): 226-234.

[14] Ricardo Baezar-Yates, Berthier Ribeiro Neto, "Modern Information Retrieval" *ACM press, New York, 1999.*

[15] M. Porter. An algorithm for suffix stripping. *Program*, 14(3): 130-137, 1980.