Multilingual Data Management System

Dr. A.S.M. Latiful Hoque, Kazi Saidul Hasan[†], Chowdhury Sayeed Hyder[‡]

Associate Professor, Department of Computer Science and Engineering, BUET, Dhaka, Bangladesh.

[†]Member, Technical Staff, Banglaphone Ltd, Dhaka, Bangladesh.

[‡]Member, Technical Staff, Banglaphone Ltd, Dhaka, Bangladesh.

asmlatifulhoque@cse.buet.ac.bd, saidul29@yahoo.com, deeyas017@yahoo.com

Abstract

Development of systems based on one language makes the Internet domain-specific and hampers internationalization. This paper presents a three-layer architecture that facilitates retrieving of any type of data composed or indexed in one language via a query formulated in another language. In this architecture, a conversion engine (middle layer) sits in between the client (top layer) and the original information repository (bottom layer). A dictionary based pipelined word-by-word conversion method has been presented that performs query and corresponding result transformation with the help of a multilingual database. This method considers the domain of the data and carries out the corresponding domain-specific conversion policy. The multilingual database assisting the conversion can be extended to provide support for new languages. The client and the conversion engine follow an XML-based communication protocol. We have considered Bengali and English language for the development purpose.

Keywords: dictionary-based approach, multilingual information retrieval, name conversion, query translation, XML-based protocol.

I. INTRODUCTION

With the great advent of the World Wide Web, the issue of multilingual information access and retrieval has surfaced. Issues relating multilingual access and multinational management of data are utterly decisive for ensuring the full benefits of the global cyberspace. So far, in the Digital Library (DL) sector, most research and development activities have concentrated on monolingual environments and, in the large majority of cases, the language employed has been English. But now both Asia and Europe are now actively involved in building up their own large distributed organized repositories of knowledge in their native languages. At present, 34.7% users use English language in the Internet. But about 40% users prefer languages like Chinese, Japanese, German, Spanish and French on the Internet [1]. If the developers of these digital libraries stick to the old and obsolete idea of monolingual interface, then ensuring global access to this information would be impossible. So, it is vital to implement a multilingual interface for a digital knowledge repository to increase the world-wide potential for access to knowledge.

II. PROBLEM DEFINITION

For an Internet user who wants to roam in the cyberspace using a language other than English, there are a few things to ensure. The user has to look for a web site providing an interface in the user's language and also the backend database should be compatible with that language. Problems arise when a user wants to retrieve information in the language of his/her choice but the repository is built on a different language. So the problem is to design and implement an interface that sits in between the user and the backend data repository, accepts queries from the user in the language of his/her choice, transforms the query to a form compatible with the backend database, queries the database and finally provides the user with the desired information in his/her language ensuring that multiple copies of the same content is not required.

III. OBJECTIVES

Our objective is to address the issues related to the development of multilingual support for a system. The architecture of the system is described with the conversion engine sitting in between the client and the database. The structure of different modules (medium dependent and medium independent) of this engine has been described. A dictionary-based approach to the problem of designing such an interface has been adopted. Issues relating to incorporating a new language in the engine and the communication protocol between the client and the engine have also been discussed.

IV. LITERATURE REVIEW

Most approaches translate queries into the document language and then perform monolingual retrieval. There are three main approaches in multilingual information retrieval (MLIR) and cross language information retrieval (CLIR): using machine translation (MT), a parallel corpus, or a bilingual dictionary. MT-based approach uses existing MT techniques to provide automatic translation of queries. The MT-based approach is simple to apply, but the output quality of MT is still not very satisfying. Corpus-based approach analyzes large document collections (parallel or comparable corpus) to construct a statistical translation model. The performance relies largely on the quality of the corpus. Also, parallel corpus is very difficult to obtain, especially for western and oriental language pairs. However, this technique tends to require the integration of linguistic constraints, because the use of only statistical techniques by extracting information can introduce errors and thus achieve bad performance [2]. In a dictionary-based approach, queries are translated by looking up terms in a bilingual dictionary and using some or all of the translated terms. This is the most popular approach because of its simplicity and the wide availability of machine-readable dictionaries. By using simple dictionary translations without addressing the problem of translation ambiguity, the effectiveness of CLIR can be 60% lower than that of monolingual retrieval [3]. We used bilingual dictionary approach to convert query to English.

V. SYSTEM ARCHITECTURE

The Multilingual Data Management System is based on a three-layer application model. Fig. 1 shows the high-level view of the system's architecture. The first of these three layers is comprised of the client program that provides the user interface to the users and requires multilingual support. This layer includes the business logic specific to the application and it contains no multilingual conversion logic. The user interface must be designed to ensure that the user can view the extracted information in appropriate format and also interact with the system using the language of his/her choice.

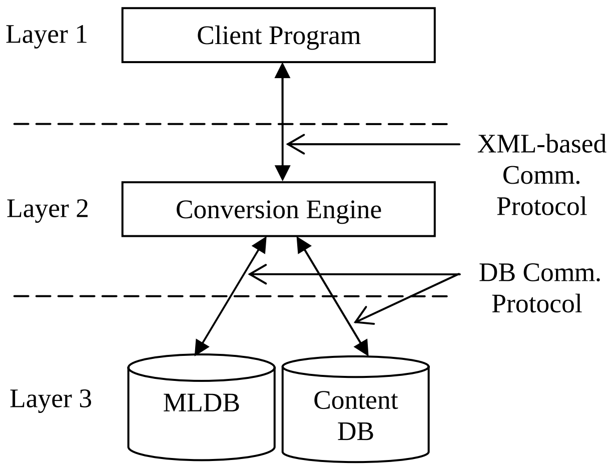


Fig. 1 High-level view of the three layer architecture of the Multilingual Data Management System.

The second layer acts as a bridge between the user interface and the database. This layer consists of a conversion engine. This engine is independent of the application that requires multilingual conversion. All queries for data pass through this engine before reaching the database. Likewise, responses travel back through the conversion layer before reaching the client. This layer accepts queries from the client, performs multilingual conversion of the query and then queries the database with the converted query. The retrieved data undergo multilingual conversion once again by the engine so that the user can see the results of his/her query in his/her chosen language.

The third layer is comprised of a database (Content DB) which is the repository of all information of the user's interest. The data layer also includes the Multilingual Database (MLDB) which is used for multilingual conversion. Database access is done in the usual fashion by the conversion engine with the help of database communication protocol (e.g. jdbc). The Multilingual Database comes into play when conversion is required of the user queries that are posted to the middle layer and also when the response of the query is to be converted before being delivered to the user by the middle layer. This database contains different language modules that contains related conversion logic and assists the conversion engine by providing these conversion rules.

VI. SYSTEM MODULES

The conversion engine maintains some component modules each of which deals with a specific domain of data. For some data, bilingual conversion is not possible. For example, name conversion has to deal with pronunciation and spelling of different languages. It widely varies from language to language. The dependency of language can be solved by introducing a language as the medium of conversion. We considered English as the medium language in our implementation. The modules used in this purpose are termed as medium dependent modules. Other domains for which one to one language mapping is possible without the involvement of a medium language are under medium independent modules.

The storage, maintenance and applicability of the medium dependent modules to translate data are very much dependent on the context of data, the internal organization of the language and bilingual transformation. Pronunciation rules of a language are mapped to English. Rules have been implemented to translate names from Bengali to English and vice versa. We considered pronunciation of vowel and consonants differently. Random sets of examples of names were taken and rules were derived by recognizing their pronunciation patterns. As the number of vowels and consonants between these two languages (Bengali and English) differs, perfect conversion is very hard to ensure. In different positions of a word, a vowel has varying pronunciation. Single or consecutive presence of vowel in a word also varies the pronunciation. Similarly, consecutive presence of consonants also affects the pronunciation. We considered three cases in the implementation of medium dependent module. These are single vowel/consonant, double vowel/consonant and relative vowel position in a word.

Medium independent module consists of three sub modules- numeric module, word module and phonetics module. As the basic digits are common to all languages, numeric module maintains a numeric mapping of digits for all languages. The word module is mainly a multilingual dictionary. Words of each language are stored in a separate Trie structure. Each word is given a unique ID. Word IDs of a language are matched against the word IDs of other languages to maintain the multilingual dictionary (many-to-many relationship). Finally, phonetics module stores phonetics of words of all languages and it facilitates better conversion of name. Phonetics of words is stored following the same procedure that is deployed to store words and a link is maintained between them.

VII. STEPS OF THE CONVERSION PROCESS

To provide the user with multilingual information retrieval (MLIR) support, some definite steps are to be followed. At first step, the user submits the query in his/her preferred language and specifies the language. The client program tags multilingual parts of the query using XML. Then the multilingual parts of the query are translated into the target language by the conversion engine. This translation step tends to cause a reduction in cross-language retrieval performance as compared to monolingual information retrieval. There are four different translation options: translating queries [4], translating documents [5], translating both queries and documents, and cognate matching. Our design is based on query translation. The translation of queries is inherently difficult due to the lack of a one-to-one mapping of a lexical item and its meaning. This creates lexical ambiguity. Also, query translation is complicated by the cultural differences between language communities. These two translation issues create many different translation problems such as lexical ambiguity, lexical mismatches, and lexical holes. After a target language-compatible query is built, it is passed to a search module of the system. After the search, the resultant dataset is converted into the target language.

To translate data with maintaining its original meaning and applicability, the domain of the dataset must be known. For example, when considering a name of a person or an organization the conversion procedure must be different from the one when considering words or phrases. Different data types require different methods to be applied for conversion. Some data may consist of multiple domains. All these factors tend to reduce the performance and efficiency of the system. Measures are taken to reduce performance deficiency.

VIII. IMPERATIVE SYSTEM FEATURES

A. Server Metadata

It is not possible to know the domain of data from database. But as the domain of data is necessary for conversion, this information should be provided by the data repository. The content providers have to provide an extra database for this purpose. This database contains domain information of the content. For example, a table column can hold data of word, numeric, name or mixed domain. The conversion is based on the data context and domain. The data domain is resolved on the basis of metadata provided by the server.

B. Translation of Word

As there is a multilingual dictionary for words, conversion of words between any two languages is easy. Appropriate word is searched from the wordlist of source language and using its ID the corresponding word in the target language is found. A word can have multiple meanings. Some client applications may need only one meaning while other systems (e.g. search engines) may require all the meanings. Our system will provide the first meaning found in the dictionary but for improved system performance, statistical methods can be applied to find the most suitable meaning.

C. Conversion of Name

Name conversion is carried out in two steps. At first, the input record is tokenized and each token is handled separately. For each token the wordlist is searched and if a match is found then associated phonetics is taken. This helps to obtain a better representation of the name in the target language. If the token is not found in the wordlist then it is passed to the name module. Relative position of letters and position in a token are verified. Using the rules derived from a set of observations, a name is converted.

D. Extensible Nature

The system is designed in such a generalized way that it can be extended to include new languages. Any language to be included must be provided with few component modules. A bilingual dictionary is to be provided. The dictionary must correspond to a language that has already been included in the system. The wordlist is provided in XML format. The provided wordlist is used in mapping with existing wordlist of different languages. Also rules for name conversion are also provided. For example, if French is to be included in the system, a bilingual dictionary along with phonetics and conversion rules of name to English is provided.

IX. THE CLIENT-CONVERSION ENGINE COMMUNICATION PROTOCOL

Since the layers exchange structurally strict data between themselves, data that is required to be transported across the client/server interface is tagged with XML so that the other party can easily manipulate XML's structural and semantic information. During the database connection initialization phase, the client just sends an

XML document to the server which contains the database name, user name, password and other connection related parameters. This part of the client/server conversation is language independent and has nothing to do with the multilingual aspects of the system. This also

```
<Query>
        <Plain_Text>... </Plain_Text>
        <Multilingual_Text>
                <Language>...</Language>
                <Text>...</Text>
        </Multilingual_Text>
        <Plain_Text>... </Plain_Text>
        <Multilingual_Text>
                <Language>...</Language>
                <Text>...</Text>
        </Multilingual_Text>
</Query>
```

Fig. 2 XML file format – Query Posting.

```
<Query_Result Status=successful>
        <Columns Column_Count=...>
        <Column_Name>...</Column_Name>
        .
        .
        </Columns>
        <Resultset Row_Count=...>
                <Row>
                        <Row_Text>
                                ....
                        </Row_Text>
                        .
                        .
                </Row>
                .
                .
        </Resultset>
</Query_Result>
```

Fig. 3 XML file format – Engine response when the query is successful.

holds true when it comes to terminating the database connection. After the connection is made, it’s the client’s turn to submit queries to the server which can include multilingual text. Here the strength of XML comes into play. Multilingual texts (e.g. text in French) are tagged separately and attributes are provided to resolve the language. This information allows the middle layer to perform the conversion (e.g. French to English) using the correct language dependent module (e.g. French-English module). After the conversion is made, the query is submitted to the database in the usual manner using the database communication protocol. The result from the database undergoes the opposite conversion (e.g. English to French). The converted database response is tagged with XML and delivered to the client. Fig. 2 and 3 shows the XML templates for query posting and successful database response.

X. SOFTWARE ARCHITECTURE

The software architecture for this multilingual information retrieval system is designed as a pipelined structure. Hence, successively activated pipeline elements apply transformations on client queries that are posted via arbitrary client devices, such as, for instance, web browser, PDA or mobile phone. Due to the flexibility of this approach, different pipelined layouts can be used to implement different processing strategies. Fig. 4 depicts the layout of the software architecture and illustrates the way of interaction of the pipeline elements.

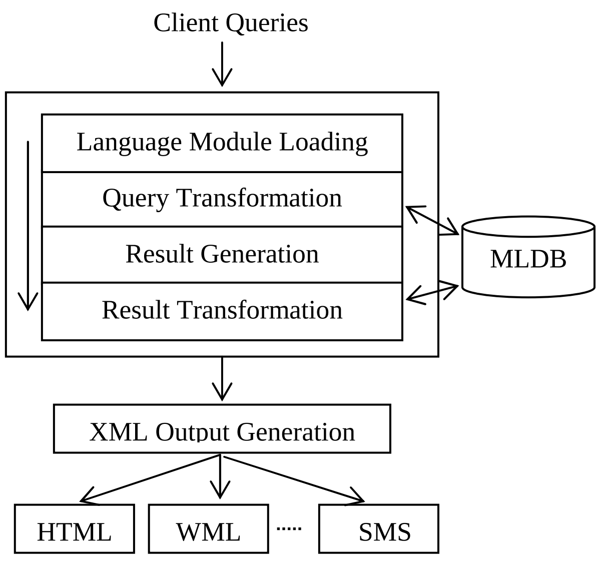


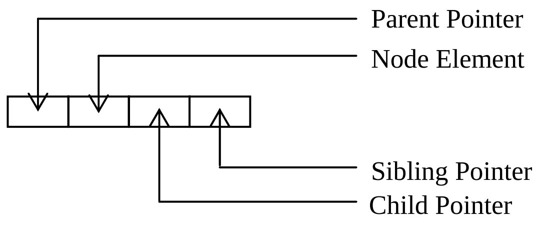
Fig. 4 Software architecture of Multilingual Data Management System.

First of all, the user query is taken as input from the client device. Then the XML script is examined to identify the language corresponding to the multilingual parts of the query. Then the particular language module is loaded. After that, the part of the query formulated in that language undergoes conversion with the help of the language module gathered from the multilingual database (MLDB) and as a result the query suitable for posting to the content database is formed. Then the database is queried and the result is obtained. This result is again transformed into the language the original user query was posted in. Then an XML representation is generated to fit the needs of the client device.

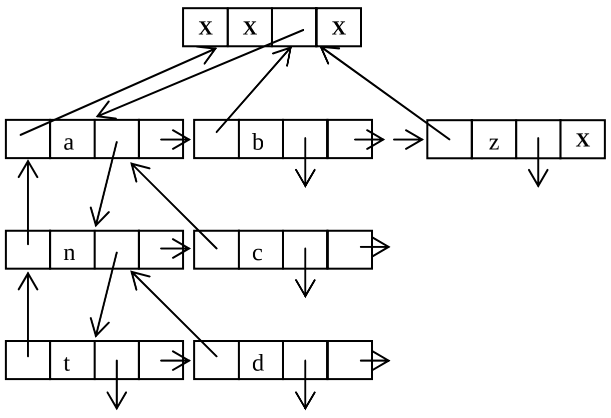
XI. DATA STRUCTURE FOR THE DICTIONARY: TRIE

For storing the dictionary nodes (words) more efficiently, the Trie structure has been used. It helps saving up a lot of space. Fig. 5(a) and 5(b) shows a Trie node structure and the overall Trie structure respectively. A

Trie node stores a single letter in the Node Element field and three pointer fields.



(a)



(b)

Fig. 5(a) Trie node structure (b) Trie structure.

Let us consider an example. For the word ‘ant’, we will build a tree of 3 levels where the letter ‘a’ is at level 1. Letters ‘n’ and ‘t’ are at level 2 and 3 respectively. The node for ‘t’ at level 3 holds a flag indicating the end of a word (ant). Now if a word starting with ‘a’ shows up then we’ll not use a new node at level 1 for ‘a’ since we already have one node for ‘a’ at level 1 (for ant). Thus all words starting with ‘a’ will share the same level 1 node for ‘a’. If a new word starting with a letter other than ‘a’ arrives (e.g. computer) then we have to create a new level 1 node. So the bottom line is, at each level, a node corresponding to the current letter in the word is looked for. If the node exists then it is used by the current word, otherwise a new node is created. For maintaining a bilingual dictionary, we must have two Trie structures, one for each language. The node where a word ends can store information like the phonetics, the word’s origin etc. Each word in the Trie structure has been assigned an ID which is used to find its equivalent in the other Trie structure (for bilingual dictionaries).

XII. ACHIEVEMENTS

In this paper, we have come up with the design of a MLIR system, corresponding software architecture and the XML-based communication protocol. New modules like spell checking module, numeral conversion module etc. can be plugged into the pipelined software architecture. The Trie structure ensures space-efficient storage of the dictionary. The name conversion module works fine for common Bengali names but more specific rules are needed for ensuring better conversion. The proposed procedure can be applied in web searching. For instance, search engine performs its operation on keywords and provides results in the order of rank. Search engines can make use of this multilingual conversion system to search multilingual documents. Following the proposed strategy, result with the same ranking can be obtained.

XIII. DISCUSSION

We couldn’t manage a machine readable Bengali-English dictionary. So we were constrained to build up a dictionary with very limited vocabulary in a manual fashion and it was time consuming. For the problem of name conversion, since we had no definite set of rules, we had to apply some rules of our own being derived from a number of observations. The performance of the proposed system can be increased by eliminating problems like translation ambiguity, presence of multiple word meanings, absence of special vocabulary and technical terminology. Techniques like phrasal translation, co-occurrence analysis, and query expansion can be used to resolve these issues. Also, retrieval from Trie structure can be made faster using the frequency count of a letter.

REFERENCES

[1] http://www.internetworldstats.com/stats7.htm.

[2] H. Haddouti, “Multilinguality Issues in Digital Libraries,” Proceedings of the EuroMed Net'98 Conference Nicosia, March 3-7, 1998.

[3] Ballesteros, L. & Croft, B. (1998), “Resolving Ambiguity for Cross-language Retrieval,” *SIGIR’98*, Melbourne, Australia, August 1998, pp. 64-71.

[4] Ballesteros, L. & Croft, B. (1996), “Dictionary Methods for Cross-Lingual Information Retrieval,” in Proc. of the 7th DEXA Conference on Database and Expert Systems Applications, Zurich, Switzerland, September 1996, pp. 791-801.

[5] Oard, D. and Hackett, P. (1998), “Document Translation for Cross language Text Retrieval at the University of Maryland,” In: Proceedings of the 6th Text REtrieval Conference (TREC-6); 1997 November 19-21; National Institute of Standards and Technology (NIST), Gaithersburg, MD. 687-696.