NLP TOKEN MATCHING ON DATABASE USING BINARY SEARCH

Ekta Agrawal
Department of of Computer Science & Engineering
Oriental Institute of Technology, Bhopal

Shreeja Nair
Department of of Computer Science & Engineering
Oriental Institute of Technology, Bhopal

ABSTRACT

Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things. The paper deals with the concept of database where by the data resources data can be fetched and accessed accordingly with reduced time complexity. The retrieval techniques are pointed out based on the ideas of binary search. A natural language interface refers to words in its own dictionary as well as to the words in the standard dictionary, in order to interpret a query. The main contribution of this investigation is addressing the problem of improving the accuracy of the query translation process by using the information provided by the database schema.

Keywords

Binary search (BS), natural language interface (NLI), database management system (DBMS), structure query language (SQL), database (DB).

1. INTRODUCTION

With the increasing pervasiveness of computers in society, there has been an upward trend in the search for human-computer interfaces for database (DB) that the generality of end-users find easy to use. A major reason for end-user difficulty with database system (DB) can be traced to the types of functionality supported by interfaces for formulating queries, i.e., to the use of expressions and other constructs that are far removed from the user's natural view of his or her application domain. There is a need for DB languages to use concepts that are as close as possible to those in the user's world. And they also need to have some idea of what the database contains and what types of natural language queries are covered. Databases are by definition closed domain systems. In natural language interface user can access data with the help of natural language then searching will be done with the help of binary search. Be creative in our designs and let us know how things turn out. Binary search is an extremely efficient algorithm when it is compared to linear search in binary search we first compare the key with the item in the middle position of the array. If there's a match, we can written immediately. If the key is less than the middle key, then the item sort must lie in the lower half of the array; if it's greater then the item sort must lie in the upper half of the array. So we repeat the procedure on the lower (or upper) half of the array. SQL commands can be use interactively as a query language, or they can be embedded in application program thus SQL is not a programming language; rather it is a data sublanguage, or data access language, that is embedded in other languages. Database management systems (DBMSs) have been widely used because of their efficiency in storing and retrieving data. However, traditional query language, as a standard database user interface, has often frustrated database user including expert users by enforcing rigidity and preciseness in query writing and returning query results exactly what are being asked. Such a direct query answering method requires the user to have a detailed knowledge of both the database schema and the query language and the user also has to know exactly what information to search. This method is inconvenient for non-experienced and casual users because they often cannot construct a query intelligently and properly. So we use natural language interface to provide convenience to access data from DB.

As an example, suppose there is a set of data comprising of the amount of milk consume by a person from this given data information can be retrieve as follows:

1. What is a total amount of milk consumed?
2. In which day maximum milk consumed?
3. In which day minimum milk consumed?
4. What is the average amount of milk consumption per day?
5. What amount of carbohydrate is assimilated?

This information is very difficult to access data from database by using SQL commands because a general user is not aware about SQL queries. So we can use NLP that accept query in natural language and translate it into SQL query and generate output in natural language.

2. LITERATURE REVIEW

2.1 Lunar

The system LUNAR [7] is a system that answers questions about samples of rocks brought back from the moon. The meaning of systems' name is that is in relation to the moon. The system was informally introduced in 1971. To accomplish its function the LUNAR system uses two databases; one for the chemical analysis and the other for literature references. The LUNAR system uses an Augmented Transition Network (ATN) parser and Woods' Procedural Semantics. According to [6], the LUNAR system performance was quite impressive; it managed to

handle 78% of requests without any errors and this ratio rose to 90% when dictionary errors were corrected. But these figures may be misleading because the system was not subject to intensive use due to the limitation of its linguistic capabilities.

2.2 Ladder

The LADDER system was designed as a natural language interface to a database of information about US Navy ships. According to [4], the LADDER system uses semantic grammar to parse questions to query a distributed database. The system uses semantic grammars technique that inter-leaves syntactic and semantic processing. The question answering is done via parsing the input and mapping the parse tree to a database query. The system LADDER is based on a three layered architecture. The first component of the system is for Informal Natural Language Access to Navy Data (INLAND), which accepts questions in a natural language and produces a query to the database. The queries from the INLAND are directed to the Intelligent Data Access (IDA), which is the second component of LADDER. According to [3], the INLAND component builds a fragment of a query to IDA for each lower level syntactic unit in the English language input query and these fragments are then combined to higher level syntactic units to be recognized. At the sentence level, the combined fragments are sent as a command to IDA. IDA would compose an answer that is relevant to the user's original query in addition to planning the correct sequence of file queries. The third component of the LADDER system is for File Access Manager (FAM). The task of FAM is to find the location of the generic files and manage the access to them in the distributed database. The system LADDER was implemented in LISP. At the time of the creation of the LADDER system was able to process a database that is equivalent to a relational Data base with 14 tables and 100 attributes.

2.3 Chat-80

The system CHAT-80 [6] is one of the most referenced NLP systems in the eighties. The system was implemented in Prolog. According to [2], the CHAT-80 was an impressive, efficient and sophisticated system. The database of CHAT-80 consists of facts (i. e. oceans, major seas, major rivers and major cities) about 150 of the countries world and a small set of English language vocabulary that are enough for querying the database. The system translates the English language question by the creation of a logical form as processes of three serial and complementary functions where:

1. Words are represented by logical constants.
2. Verbs, nouns, and adjectives with their associated prepositions are represented by predicates. The predicates can have one or more arguments.
3. Complex phrases or sentences are represented by conjunctions of predicates. These functions are being; parsing, interpretation and scoping. The parsing module function determines the grammatical structure of a sentence and the interpretation and scoping consist of various translation rules, expressed directly as Prolog clauses. The basic strategy followed by Chat-80 is to append some extra control information to the logical form of a query in order to make it an efficient piece of Prolog program that can be executed directly to produce the answer. According to [6], the generated control information comes into two forms:

1. Orders the predications for a query that will determine the order in which Prolog will attempt to satisfy them.
2. Separates the over all program into a number of independent sub problems to limit the amount of backtracking performed by prolog.

In this way, Prolog is led to answer the queries in an obviously sensible way and the Prolog compiler can compile the transformed query into code that is as efficient as iterative loops in a conventional language.

3. MOTIVATION

Dictionary are often used to store large amount of data from which individual words must be retrieved according to some such criterion. Dictionary can be collection of words and suppose user can write a query then individual word can be search on that dictionary if the all words are found on dictionary that means the word is valid and query generator generate the corresponding SQL statement for this natural query by using semantic dictionary.

In previous system data is stored in unsorted form so we can search words from lexical dictionary and production rules from semantic dictionary we apply linear search method to search the words from both the dictionaries.

In linear search word can be search by sequentially moves through your collection (or data structure) looking for a matching value.

3.1 Algorithm for Linear Search

Let D be a lexical dictionary of n elements D(1),D(2),D(3)……D(n). "Word" is the element to be searched. Then this algorithm will find the location "LOC" of word in D. If the word is found on that location then word is valid otherwise, word is invalid.

1. Input in dictionary D of an element "word" to be searched and initialize LOC=-1.
2. Initialize i=0; and repeat through step 3 if(i<n) by incrementing i by 1.
3. If (word=D[i])
   (a) LOC=i
   (b) GOTO step 4
4. If (LOC>0)
   Print " Word is found and searching is successful".
   Else
   Print "Word is not found and searching is unsuccessful".
5. Exit

If we search the word on dictionary by using linear search method, it is a very time consuming method words are stored in unsorted form so if want to search a particular word on a dictionary speed of searching is very slow, data can be stored in a sequential manner on a continuous location in a dictionary.

4. PROPOSED WORK

1. In this we propose the concept of binary search for storing word in dictionary.
2. In this concept words are stored in block form and that blocks are arranged in sorted manner.
3. Then we apply binary search on that block.
4. It reduce the searching time in comparison to conventional dictionary.
5. Words can also be add easily in dictionary.

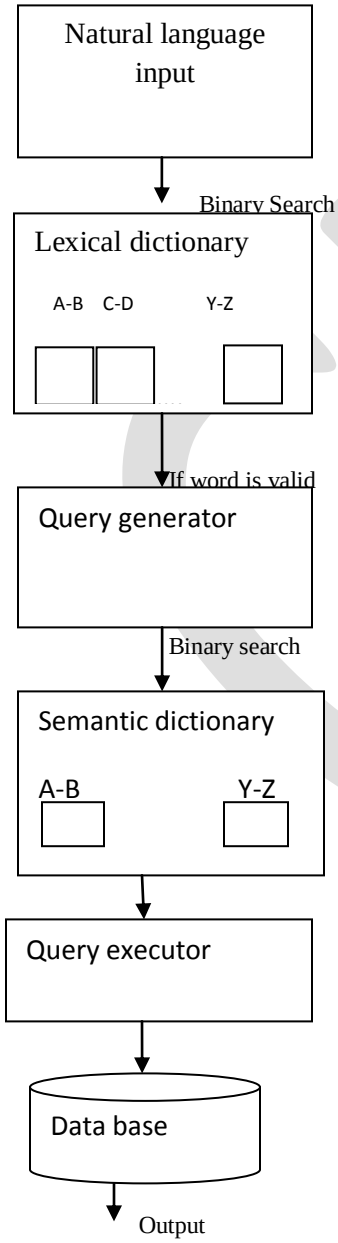

Figure: Flowchart for "NLP TOKEN MATCHING ON DATABASE USING BINARY SEARCH"

4.1 Natural Language Input

User can interact with our system in the form of natural query. There is no need to learn any DB query language to access data from database. In natural language interface user can interact easily with the system by using natural language English.

4.2 Lexical dictionary

In lexical dictionary words are arranged in blocks .these blocks are arranged in alphabetical order .and words on that blocks are also in alphabetical order.

If the user write a query in natural language then individual words are search in lexical dictionary by using binary search method.

4.3 Algorithm for Binary Search

1. Algorithm for binary searching (d,n,x)
2. //d is an dictionary of n elements.
3. // x(word) is an element we want to search
4. Begin
5. Left=0
6. Right=n-1
7. Flag=False
8. While(Left<=Right)do
9. {
10. mid=[(Left+Right)/2]
11. If(x<d[mid])then
12. Right=mid-1
13. Else if(x<d[mid])then
14. Left=mid+1
15. Else return mid
16. Flag=True
17 End if
18. End if
19.}
20. End

4.4 Query Generator

If the word is valid in lexical dictionary then query generator generate the corresponding SQL query by using semantic dictionary.

4.5 Semantic Dictionary

In semantic dictionary words are arranged in blocks .these blocks are arranged in alphabetical order .and words on that blocks are also in alphabetical order.

This dictionary consists of the English semantic grammar (grammar rules) and the schema of the database in use. The semantic dictionary is used to replace words and/or phrases semantically by equivalent words and/or phrases that are recognized by our system (according to the system capabilities).

These dictionary is also arrange in a sorted manner so we apply binary search method for searching a production rules or grammar for a specific word.

4.6 Query Executor

The purpose of Query executor is to get the required results from the used database. In order to achieve this, the generated SQL statement would be tested to verify correctness before applied to the used database and then represent the result to the user.

5. CONCLUSION

In this paper we propose the concept of binary search in natural language processing with this concept:

1. We reduce the time of processing to search the word in dictionary.
2. Output will be generate very fast.
3. Words are arrange in a blocks so the efficiency of system will be increase.

6. REFERENCES

[1] Faraj A. El-Mouadib, Zakaria S. Zubi, Ahmed A. Almagrous, and Irdess S. El-Feghi "Generic Interactive Natural Language Interface to Databases (GINLIDB)" INTERNATIONAL JOURNAL OF COMPUTERS Issue 3, Volume 3, (2009).

[2] T. Amble, "BusTUC - A Natural Language Bus Route Oracle." 6thApplied Natural Language Processing Conference, Seattle, Washington,USA, 2000.

[3] G. Hendrix, "The LIFER manual A guide to building practical natural language interfaces", SRI Artificial Intelligence Center, Menlo Park,Calif. Tech. Note 138, 1977.

[4] G. Hendrix, E. Sacrdoti, D. Sagalowicz, and J. Slocum, "Developing a natural language interface to complex data", ACM Transactions on Database Systems, Volume 3, No. 2, USA, 1978, pp. 105 – 147.

[5] D. Warren and F. Pereira, "An efficient and easily adaptable system for interpreting natural language queries in Computational Linguistics" Volume 8, 1982, pp. 3 – 4.

[6] W. Woods, "An experimental parsing system for transition network grammars. In Natural Language Processing", R. Rustin, Ed., Algorithmic Press, New York, USA, 1973.

[7] W. Woods, R. Kaplan and B. Webber, B. "The Lunar Sciences Natural Language Information System", Final Report. B. B. N. Report No 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, USA, 1972.