Efficient Transformation of a Natural Language Query to SQL for Urdu

Rashid Ahmad, Mohammad Abid Khan and Rahman Ali
Department Of Computer Science,University of Peshawar Pakistan
rashidahmad_uop@yahoo.com, abid_khan1961@yahoo.com, rahmanali.scholar@gmail.com

Abstract

It is a long term desire of the computer users to minimize the communication gap between the computer and a human. Natural Language Interfaces to Databases (NLIDBs) is one of the mechanisms to pull off this goal. In NLIDBs the question is asked in simple daily life human language and the answer is given in the same language. This research paper is about NLIDBs for Urdu language. An algorithm is developed that efficiently maps a natural language query, entered in Urdu, to an SQL (Structured Query Language) statement. The algorithm has been implemented in Visual C#.NET and tested on a database containing Student Information System and Employee Information System. The program correctly maps 85% natural language queries.

1. Introduction

Natural Language Interfaces is a hot area of research since long. Asking questions from a database in natural language is a user friendly way of searching databases rather than writing and posing a question in the restricted pattern of SQL syntax. Although the nature of questions and vocabulary for a particular natural language interface is limited in some way but the user is more comfortable in writing questions in natural fashion instead of learning the keywords and syntax of the SQL.

The success of designing Natural Language Interfaces to Databases (NLIDBs) are partly because of the real world payback of the field and partly because Natural Language Processing (NLP) works well in a particular database domain [1]. A number of researchers have developed different NLIDBs. Most of the early systems are based on pattern matching [2]. Lunar was a natural language based query system that answered questions about rock samples brought back from the moon [1]. This system was able to answer 90% of the questions in its domain when posed by untrained people [2].

LADDER was the first semantic grammar-based system, interfacing a database with information on US Navy ships [2]. Semantic grammars are now widely used in most NLP systems [1]. A semantic grammar is a formal definition of a language that uses concepts from a particular domain of discourse to specify acceptable expressions in that language [3].

A large part of the research in the middle of eighties was devoted to portability issues [1]. An example of this kind of system is TEAM [4]. TEAM was the result of a four years project and the core endeavor behind it was to design a portable NLIDB instead of the one that is domain specific. The design decisions incorporated in TEAM were generally applicable to a wider range of natural-language processing systems [4]. However for some of the systems, TEAM was forced to take a more limited approach.

STEP is a natural language interface to relational databases developed by Michael Minock [5]. It is also based on semantic grammar and uses paraphrasing mechanism to treat the natural language query. Moreover, it is relatively trouble-free to configure for domain specific databases.

Some work has also been done on the theoretical model of representing English sentences in Prolog [6]. The restraint of the work is similarity of the sentences that had been taken for examples.

Semantic grammars are mostly used these days in the design of NLIDBs.. An example of such a recent work is PRECISE [7]. PRECISE is a system that guarantees the correct mapping of a natural language query to an SQL statement, if a query is semantically tractable. Moreover, the system is also proficient in resolving ambiguities that arise due to the possibilities of a value token for multiple columns. For example, a particular database could contain the value HP under a column company and also under a column platform.

This work is about the transformation of a natural language query in Urdu to SQL. The proposed algorithm efficiently maps a semantically tractable natural language query to an SQL statement. The system is based on formal semantics like PRECISE, but a more efficient approach has been taken to deal with the value tokens. The proposed approach does not look for the value tokens in the database as in PRECISE, instead, it uses the AV Mapping algorithm to map a particular value token to its corresponding attribute token.

In the rest of the paper, section 2 discuses the requirements of a natural language interface to databases. It describes all the pros and cons that were required to put up this natural language interface for Urdu language. Section 3 highlights the layout and working of the attribute/value mapping algorithm and its efficiency in transforming the natural language query to an SQL statement. Section 4 is about query transformation and the intermediate representation of the query. Section 5 gives the implementation details and its corresponding results that were experimented during its testing. Section 6 presents the limitations. Section 7 sketches some future work dimensions.

2. Considerations of the constructed Natural Language Interface

For minimalism and understanding, this section is divided into seven sub-sections. They are listed below.

1. Types of questions.
2. Tokens formulation
3. Syntactic Markers
4. Extraction of necessary parameters
5. Main Keywords
6. Construction of Dictionary
7. Structure of Semantic Knowledge Base

2.1. Types of questions

NLIDBs are designed to dig out the information from the database using a query in a natural language. The natural language queries for Urdu language are divided in two categories. The query posed by the user will be either a question or a request. For the proposed system to work, the query must match one of the given categories. The words used for request or questions are the basis for the extraction of the required parameters from a natural query. Besides, the positions of these words in a query have a vital role in identifying the type of the parameter. These parameters could be the name of a table, attribute or value. On the whole, the mentioned three parameters are required in an SQL statement to work out properly. A simple example of each type of query is given to reveal the nature of queries.

(1) مجھے امجد کی کلاس کا نام بتایں-

[mʊʤeI] [ʌmjəd] [ki] [kɒlɒs] [kɒ] [nɒm] [bɒtɒyæñ]
[To me] [Amjad] [of] [class] [his/her] [name] [tell]
"Tell me Amjad's class name."

(2) امجد کی کلاس کونسی ہے؟
[ʌmjəd] [ki] [kɒlɒs] [kəʊnsi] [hæ]
[Amjad] [of] [class] [what/which] [is]
"Which one is Amjad's class?"

Example (1) is a query of type request and example (2) is a query of type question.

2.2. Tokens formulation

To make a sentence able to be processed by the computer, it is necessary to divide it in chunks or tokens to understand its meaning and structure. In tokens formulation, the sentence is divided in small chunks known as tokens. In Urdu, the words are separated by space as it is in English except some compound words that is not the concern of this paper. The proposed system tokenize a sentence into small pieces or tokens, which then undergo for further processing for other steps.

For simplicity and efficient transformation, we have given an order number to each of the token category. This ordering makes it effortless to identify the category of tokens for further processing. For example, if a category is of no meaning for further operation, that category is ignored for further processing.

2. 3. Syntactic markers

This NLIDB is based on formal semantics and deals a natural language query semantically. It is necessary to define and ignore the syntactic markers for further processing as they do not have semantic contribution in tracking a query semantically. A syntactic marker (such as "the") is a token that belongs to a fixed set of database-independent tokens that makes no semantic contribution to the interpretation of the question [7]. For example, a simple query in Urdu is:
(3) اسلم کے والد کا کیا نام ہے؟
[ʌslɒm] [keI] [vɒlId] [kɒ] [kIyɒ] [næm] [hæ]
[Aslam] [of] [father] [apostrophe] [what] [name] [is]
"What is Aslam's father name?"

In this sentence the word (**کے**) is a syntactic marker. It is being the part of a natural sentence and concatenating the word (**اسلم**) to the word (**والد**), but when we necessitate translating it to SQL, it adds no contribution in the process of transformation to an SQL statement. In order to treat the natural query semantically, we ignore these syntactic markers.

2. 4. Extraction of necessary parameters

To successfully translate a natural language query to SQL, there is a need to identity the required parameters such as table name, attribute, and a value. To understand the extraction of parameters from a natural language, there is a call for that to understand the structure of the SQL. To start with, let us say Q be an SQL query, O the operator and OP the operand:

```
Q = SELECT Name FROM Personal WHERE Name
= 'ahmad'
```

By looking at the query in the given example, we can have the following conclusion. A query Q is formed of operator/s O and operand/s OP such that for each O there is a corresponding OP. In the Query Q SELECT, FROM and WHERE are the operators (O), whereas Name and 'ahmad' are the operands (OP). Its structure implies the need to extract the operands and their respective operators from a natural query. This in consequence, calls for the need of a lexicon/dictionary and a set of rules. We will look at more detail on the extraction of parameters in section 3.

2. 5. Main keywords

This section comprises the list of the keywords that are used by the parser of the system to identify the parameters and constructs. They help to design the algorithm and make it trouble-free to define the logic for identifying the types of tokens. In table 2.1 the list of keywords and their types is given.

Table 2.1 Keywords

Stop Words	Question Words	Request Words
کا (of-Masculine)	(What) کیا	رکھایں (Show me)
کی (of-Feminine)	(When) کب	بتایں (Tell me)
کے (of-Plural)	(When) کون	چاہیے (eedI n)
ہے (is-Singular)	(Where) کہاں	**Pronouns**
ہیں (is-Plural)	**Criteria Words**	مجھے I
میں (in-preposition)	جو (Who Sg/Pl)	میں I
سے (from-preposition)	جس (Who Sg)	اس (He/She)
پر (on-preposition)	جن (Who Pl)	ان (Them)

2. 6. Construction of dictionary

The aim of a natural language interface is to facilitate the user to computer in a natural way. For this purpose, we have designed a domain specific dictionary to keep the synonyms of the columns and tables names. The inclusion of synonyms makes it possible for the user to write a sentence in different natural ways. We call this dictionary as semantic dictionary, because there is no syntactic information for tokens. Rather, it will be used by Attribute Value Mapping Algorithm. For attribute value mapping, there is a detailed discussion in section 3.

Semantic dictionary contains the synonyms for each of the column and table. It also contains the plurals for each word because there is no addition of "s" or "es" as it is for English language. It is difficult in Urdu to place additional words to make a word's plural, because it requires a huge knowledge base. Table 2.2 contains some sample words with their meanings and plurals.

Table 2.2 Urdu words and their plurals

Word	Meaning	Plural	Addition of Characters
ادایگی	Payment	ادایگیاں	اں
پتہ	Address	پتے	ے
مزدور	Employee	مزدور	No Addition

Table 2.2 describes the structure of Urdu words and their plurals. As each word has a different plural, we cannot define rules to convert a word into its plural form without the need to place it in the database. We use this dictionary to resolve the names of the tables/columns from a natural language query to SQL.

2.7. Structure of semantic knowledge base

The aim of the constructed system is to track the correctness of a query semantically. For this, all the semantic information is available in the semantic dictionary that is obligatory for the process of transformation. We have designed an attribute value mapping algorithm that will efficiently transform the natural language query to SQL using the semantic knowledge base. The semantic knowledge base contains three main tables, where we have put the semantic information for a database that will be used by AV mapping algorithm. This algorithm is discussed in Section 3 in more detail. Figure 2.1 depicts the structure of the semantic knowledge base.

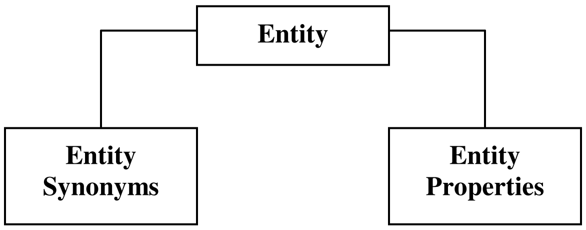

Figure 2.1 Semantic knowledge base

This dictionary is manually constructed and is database specific. The dictionary is not like a huge corpus; rather it has entries according to the number of entities in a database. There is no grave processing involved in utilizing this dictionary during query transformation, which makes this system efficient.

To start the discussion, on the storage of data within semantic dictionary, we need to understand the concept of an entity. An entity can be a person, place, thing, and concept or even about which an organization collects data [7-8]. Here are the steps that are essential to construct this semantic dictionary.

a) Identifying all the entities in a database
b) Finding out and writing the synonyms of the identified entities
c) Defining the characteristics of the entities

We will exemplify the storage of data within these tables through an example. Suppose, we have a small School Management System, and there is the need to construct the semantic dictionary for it. For the sake of understanding, let us take only a few entities from the whole system. Consider, S as the school management system, E the entities, α the synonyms and β the characteristics of an entity E. We take three entities and construct the dictionary as given in the table 2.3.

Table 2.3 (a) Entities

Entity	Meaning
طالب علم	Student
کورس	Course
جماعت	Class

Table 2.3 (b) Synonyms

Entities	طالب علم	کورس	جماعت
Synonyms	طالب		درجہ کلاس

Table 2.3 (c) Properties

Entities	طالب علم	کورس	جماعت
Properties	نام پتہ فون نمبر مضمون جماعت جنس	نام اجزاء نمبر حیثیت کل نمبر	نام تعداد بورڈ کھڑکیاں کرسیاں

This concludes the following implications from the given tables. For each entity E_i there are many synonyms (α_i…….. α_n) such that E_i = (α_i…….. α_n). Similarly, each entity has a number of properties (β_i……β_n) such that E_i has (β_i……β_n). Here are few sentences that highlight the deployment of these tables by the AV Mapping algorithm.

(a) وسیم کہاں رہتا ہے؟
[vɒsIm] [kɒha:ñ] [rəhtɒ] [hæ]
[Waseem] [where] [live] [is]
"Where does Waseem live?"

(b) پانچویں جماعت میں کتنی کرسیاں ہیں؟
[pɒntʃvñ] [ʤɒmɒt] [meIñ] [kItni] [kʊrsIyɒñ] [hæñ]
[5th] [class] [in] [how many] [chairs] [are]
"How many seats are there in 5th class?"

(c) کیمسڑی کے اجزاء کیا ہیں؟
[kəmIstɒri] [keI] [ʌʤzɒ] [kIyɒ] [hæñ]
[Chemistry] [of] [contents] [what] [are]
"What are the contents of Chemistry?"

In each of the sentences above, one characteristic of each of the three entities has been used to be evidences for how this semantic dictionary is utilized. In sentence (a) a question has been asked about the entity Student. Student has a property of living that is given in table 2.3(c) as (پتہ), which is the synonym of word (رہتا, live). This characteristic is found in the column of (طالب علم, Student), which reflects that the assumed value "Waseem" is the name of a student.

In the second sentence, the question has been asked about the number of chairs (کرسیاں) in 5th class. The characteristic chair reflects that the user is asking question about a class as this characteristic is found in the column of class in table 2.3(c). The third question asks about the contents (اجزاء) of a course or subject. As "contents" is a characteristic of a course, which implies that the user is asking question about the course. Its algorithmic details of how all this works has been shown in the next section of AV Mapping algorithm.

3. Attribute Value (AV) mapping algorithm

This section confers the AV Mapping algorithm that efficiently maps a natural language query to SQL using semantic tables described earlier. Figure 3.1 shows this algorithm.

```
//Tokenization algorithm
  Module 1 (Scanning)
  - Start Scan

        1. Split the query (Q) in tokens (t1…tn)
        2. Give an order number to each of the
        tokens identified.
            i-    Stop Words order t1
            ii-   Question or Request words
                  order t2
            iii-  Attribute/Column/Value
                  order t3
            iv-   connectors/splitters/criteria
                  words order t4
  -End Scan
```

Figure 3.1 Tokenization

The algorithm starts working by scanning the query posed by the user. The query is tokenized using tokenization rules. The identified tokens are then further processed for ordering/ranking. This ordering is done in order to simplify the work with the implication that only those order categories will be considered for further processing that shell out semantic contribution in the interpretation of the query. An important point in this algorithm is that it does not consult the database for value tokens in order to minimize and speed up the query transformation process, which is different from PRECISE. Instead, AV Mapping algorithm assumes all tokens as values that are left at the end after identifying all other tokens for their respective categories. For simplicity, AV Mapping algorithm also rank value tokens as order ranking 3, which is the same as that of an attribute, so that it should be effortless to treat attribute/value on the same level in AV Mapping algorithm. At the end, these value tokens are mapped to their respective attributes following some rules, and those tokens are ignored that contribute nothing to the query in terms of semantics.

Hence, all the tokens having order number 1 are ignored for further processing. The next important step is the identification of the type of question as explained in section 2.1. These questions are divided into two categories on the basis of their nature as discussed in section 2.1. If the question lies in neither of the category, we say the question is intractable; otherwise the question is forwarded for further processing.

Another piece of algorithm is given in Figure 3.2. In this algorithm the process of query break up and its storage in attribute/value pattern has been shown.

```
//Algorithm that splits the query
//and extracts patterns from
Module 2 (Query Splitting)
-Start Processing Order

   Formatted Query
1.  Look for the sentence
        connectors/splitters/criteria words
2.  If (word = connector(and/or))
                  Then
    queryparts = splitquery(Q)
    connectors= store connectors
         and their positions
  End If
If (word=splitter/criteria)
                  then
      Queryparts= splitquery(Q)
      Module 3(queryparts)
  Else
          Module 3(queryparts)
  End If
-End Processing
```

Figure 3.2 Pattern Extraction

Figure 3.2 shows the working of the query splitter algorithm. Once the query is scanned and unnecessary tokens are removed, we break up the query into small chunks in order to treat each chunk individually.

The query is broken down on the basis of connector tokens or splitter/criteria tokens. In Urdu language (like English), connector words/tokens are used within the queries where a user may want to ask multiple things. The sentence connector words in Urdu are (اور) and (یا), each of which respectively stands for "and" and "or".

Secondly, this algorithm also checks the query for criteria tokens. If a criteria token is found, then the second condition in the algorithm breaks up the query on the basis of this token. There is ever an attribute value pattern after a criteria token. Here examples of both of the connectors and criteria tokens are given.

(a) سلمان کی تنخواہ کیا ہے اور وہ کہاں رہتا ہے ؟
[sʌlmɒn] [ki] [tʌnxʊ:ɒh] [kIyɒ] [hæ] [ɒəʊr] [vəʊh] [kɒhɒñ] [rəhtɒ] [hæ]
[Salman] [of] [salary] [what] [is] [and] [he] [where] [lives] [is]
"What is the salary of Salman and where does he live?"

Here the connector token is اور (and). If the query is broken up on the base of اور (and) token, it results in the given sentences.

(i) سلمان کی تنخواہ کیا ہے؟
(ii) وہ کہاں رہتا ہے؟

The second example shows the use of criteria words in a sentence and their break up through the splitter algorithm.

(b) مجھے ان طلباء کے نام بتایں جو پشاور میں رہتے ہیں-
[mʊʤheI] [ʊn] [tɒlbɒ] [keI] [nɒm] [bɒta:yæñ] [ʤəʊ] [plʃɒvə(r)] [melñ] [rəhtel] [hæñ]
[to me] [those] [students] [of] [tell] [who] [Peshawar] [in] [live those] [are]
"Tell me the names of those students who live in Peshawar?"

Criteria tokens are used to specify a condition in a query, just like in the above query, جو (who) is a criterion token. In Urdu language, there will be ever a required thing before criteria token and an attribute value after it or vice versa. When this query is broken down on the basis of a criteria token, we will get the following chunks of the query.

(i) مجھے ان طلباء کے نام بتایں
(ii) پشاور میں رہتےہیں

After this is done, the chunks will be forwarded to Module 3 for further processing, where AV Mapping will get in action for each chunk of the query. The splitting of the query into chunks will provide an ease in identifying the relevant values for attributes. In the example above, the first piece of the query gives us the required information, that is the names of the people (لوگوں کے نام), and the second chunk is asking about the address (رہتے, live in) and its respective value as the name of the city (پشاور).

```
//Algorithm that will map an //assumed value for
an //identified column
Module 3 AV Mapping(queryparts)
-Start Making AV Patterns

 For each chunk ci=1 to ∑c in queryparts
   If (ci has col and has no val

        and ∑c == 1) then
           stop processing
           prompt "Intractable"
   End If
  If (ci has no col and ∑c==1)then
            stop processing
           prompt "Intractable"
    End If
        If (ci has a col and no val
               and ∑c > 1) then
               ReqColumns[x] = col
     End If
       If (ci has a col and val) then
               ReqColumns[x] = col
               AVPattern[y] = col + val
     End If
     If (ci has val and no col
        and ∑c > 1) then
      AVPattern[y]=
      ReqColumns[len – 1] + val
    End If
-End AVPatterns
```

```
//Algo takes each pair and if //necessary replace a
synonym with //proper attribute name
//SD (Semantic Dictionary)
-Start AV Mapping
  For each pattern p in AVPattern
    att = Extract att (p)
   For each characteristic c in SD
     If (c is a match for p) then
            entity = extract from SD
      If (att = synonym of c) then
         att = table_col_name
      End If
     End If
   End For
  End For
-End AV Mapping
```

Figure 3.3 Attribute value mapping

The AV Mapping algorithm has been shown in the figure 3.3 that efficiently maps the assumed values to their respective attributes. If the query is not in proper format, the value tokens fail to map to their respective attribute tokens. This results with the response for a query as intractable. By contrast, if the query lies in one of the categories either a proper question or request, the values are successfully mapped and we get an intermediate form of the query that is effortlessly transformable to SQL.

4. Query transformation

After a query is processed by the AV Mapping algorithm, it is equipped to be transformed to SQL. The AV Mapping Algorithm transforms the query into an intermediate form with the resolution of attribute names and proper binding of attribute and values. Transformation from natural language query to SQL is shown in 4.1.

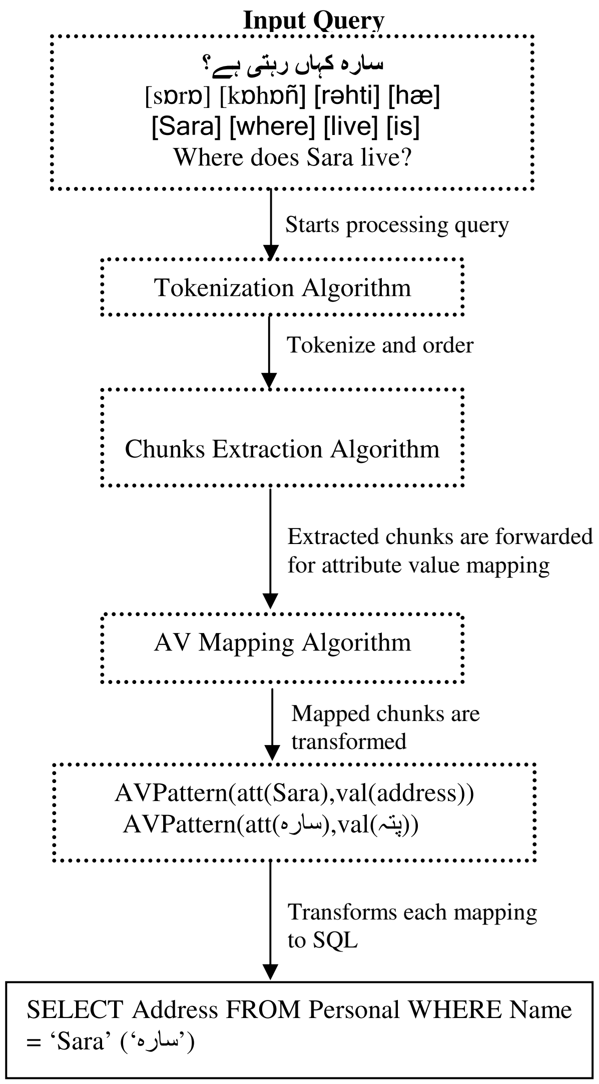
Input Query
ساره کہاں رہتی ہے؟
[sɒrɒ] [kɒhɒñ] [rəhti] [hæ]
[Sara] [where] [live] [is]
Where does Sara live?
Starts processing query
Tokenization Algorithm
Tokenize and order
Chunks Extraction Algorithm
Extracted chunks are forwarded for attribute value mapping
AV Mapping Algorithm
Mapped chunks are transformed
AVPattern(att(Sara),val(address))
AVPattern(att(ساره),val(پتہ))
Transforms each mapping to SQL
SELECT Address FROM Personal WHERE Name = 'Sara' ('ساره')

Figure 4.1 Query transformation

5. Experimental results

We have tested our algorithm on two query sets which we have collected from the users of the relevant departments. The first query set was for School Management System and the second query set was for Employee Information System. The results were quite satisfactory. The experimental results are shown in table 5.1.

Table 5.1 Experimental Results

Database Name	Questions Asked	Correctly Mapped	Accuracy Percentage
School Management System	200	172	86 %
Employee Information System	200	167	84 %

6. Limitations

The AV Mapping algorithm accurately maps most of the queries and relies on semantic dictionary to work out. It is undesirable to construct the semantic dictionary manually for every database. Our system failed to map some of the queries that were correct semantically but were not equipped with the proper information necessary for AV Mapping algorithm to map it properly to SQL. An example query is like:

(a) I need salma's marks in chemistry.

Here in the sentence "salma" and "chemistry" are the value tokens referring a student and a subject respectively, and "marks" is an attribute token. The request is asking for the student marks, but "marks" is a characteristic of a subject not of a student. To map correctly who salma is, the "marks" should be "obtained marks" instead of the general characteristic "marks". However this limitation has been addressed in the future work on the improvement of structure of semantic dictionary.

7. Future work

Research is done from the last few decades on Natural Language Interfaces. With the advancement in hardware processing power, this goal has got realization. That is, some of the NLIDBs got promising results as we mentioned in the historical background. In this research paper, we have presented an AV Mapping algorithm that accurately maps a natural language query in Urdu with minimum transformation time. As a future direction, for complex databases, we need to construct more robust ambiguity resolution algorithms. This ambiguity normally arises because of the characteristics conflicts using the semantic dictionary for more than one entity. The structure of the semantic dictionary can be improved with the consideration of mapping a complex query. There is also the need to formulize the summary, grouping (Group By, Order By), and other constructs in a natural way so that a user can get the same results using natural language. The algorithms discussed in the paper are applicable to other similar languages also, so we can work in other languages as well that have similar structure to Urdu or English using the techniques given in the paper.

8. References

[1] S. Knowles, "A Natural Language Database Interface for SQL-Tutor", 5th November 1999.

[2] V. Lopez, E. Motta, V. Uren and M. Sabou, "State of the Art on Semantic Questioning Answering", May 2007.

[3] Retrieved: (August, 4, 2008)
Available:
http://library.ahima.org/xpedio/groups/public/documents/ahima/bok1_025042.hcsp?dDocName=bok1_025042

[4] P. Martin, D. E. Applet, B. J. Grosz, F. Pereira, TEAM: "An Experimental Transportable Natural Language Interface", Artificial Intelligence Center, SRI International Menlo Park, California 94025, November 1986.

[5] M. Minock, STEP A Natural Language Interface to Database, Available: http://www.cs.umu.se/~mjm/
Retrieved: (February, 15, 2008)

[6] A. A. Sultan, *Natural Language Interfaces*, M.sc Thesis, Department of Computer Science, University of Peshawar, January 1993.

[7] A. M. Popescu, O. Etzioni, H. Kautz, "Towards a theory of Natural Language Interfaces to Databases", *University of Washington, Computer Science, January 2003.*

[8] Retrieved: (August, 21, 2008).
Available:
damacoc.org/presentations/2007_04_11_Adelman_DWGlossary.doc

[9] David H.D. Warren, Fernando C. N. Pereira, "An Efficient Adaptable System for Natural Language Queries", Artificial Intelligence Center, SRI International 333 Ravenswood Avenue Menlo Park, CA 94025, July 1982.