Proceedings of the Pakistan Academy of Sciences:
A. Physical and Computational Sciences 54 (4): 347–364 (2017)

ISSN: 2518-4245 (print), 2518-4253 (online)

Pakistan Academy of Sciences

Research Article

Identification and Resolution of Ambiguities using AV Mapping Algorithm for Query Transformation

Rashid Ahmad[1], Mohammad Abid[1], and Rahman Ali[2]

[1]Department of Computer Science, University of Peshawar, Peshawar, Pakistan
[2]Quaid-e-Azam College of Commerce, University of Peshawar, Peshawar, Pakistan

Abstract: In information technology, one predominant requirement is to design interfaces in more natural way in order to enable users to interact with computers in an easy way. Natural Language Interfaces to Databases (NLIDBs) is one of the mechanisms to achieve this goal. This paper is based on the previous work on NLIDBs for Urdu/English language in which an attribute value (AV) Mapping Algorithm was introduced. This algorithm uses semantic dictionary to efficiently map natural language queries to SQL queries and minimize its transformation time, but suffers from the ambiguity problem occurring in the queries. To overcome this problem, this study focused on the identification and resolution of ambiguous queries. New enhancement in the AV Mapping algorithm was made to improve the processing and enabling the algorithm to identify and disambiguate many ambiguous cases. All the techniques suggested in the AV Mapping Algorithm are applicable to any other language that has subject-verb-object pattern. That is, the algorithm is based on the identification and treatment of semantic tokens using semantic dictionary. The algorithm is implemented in Visual C#.NET and tested on Student Information System and Employee Information System databases. The accuracy of correct mapping is 85%.

Keywords: NLIDB, query transformation, AV mapping algorithm, SQL

1. INTRODUCTION

Natural Language Interfaces to databases (NLIDB) is about asking questions from a database in natural language in a user friendly way of searching databases rather than writing and posing a question in the restricted pattern of SQL syntax. NLIDB systems allow users to communicate with database in much the same way as people communicate with each other [1].The most commonly used methodologies include: pattern matching, representing information in formal logic (e.g. Prolog), semantic grammars, and the use of domain specific knowledge base with the semantic grammar. The attribute value (AV) mapping algorithm proposed in the paper uses semantic grammars along with a semantic dictionary. The suggested semantic dictionary contains the information about domain entities and their respective attributes. Moreover, entities and attributes are further extended with synonyms to allow the users to ask a query similar to an existing query but with different words referring to the same entity. In this way, the proposed algorithm adds more to the efficiency of query transformation process. The way in which semantic knowledge has been utilized in the proposed system is applicable to almost any human language which has a subject, verb and object pattern in the query. AV Mapping Algorithm performs the transformation process in an efficient manner using the semantic dictionary, which is novel in its structure. The algorithm skips any syntactical processing and relies on the ordering number of tokens which is assigned during the scanning or tokenization. This order numbering makes the algorithm efficient and unique in its processing from traditional NLIDB systems.

The success stories of 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 [2].

Received, June 2016; Accepted, November 2017
*Corresponding author: Rahman Ali, Email: rahmanac1@gmail.com

A number of researchers have developed different NLIDBs. Most of the early systems are based on pattern matching [3]. Lunar was a natural language based query system that answered questions about rock samples brought back from the moon [2]. This system was able to answer 90% of the questions in its domain when posed by untrained people [3].

LADDER was the first semantic grammar-based system, interfacing a database with information on US Navy ships [3]. Semantic grammars are now widely used in most NLP systems [2]. 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 [4].Semantic Analysis is related to creating the representations for meaning of linguistics inputs. It deals with how to determine the meaning of the sentence from the meaning of its parts [1].

A large part of the research in the middle of eighties was devoted to portability issues [2]. An example of this kind of system is TEAM [5]. TEAM was the result of a four years project and the core endeavor behind it was to design a domain independent 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 [5]. 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 [6]. It is also based on semantic grammar and uses paraphrasing mechanism to process 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 [7]. The restraint of the work is similarity of the sentences that had been taken as examples.

Natural language queries are ambiguous by nature. Tang [8] presented a machine learning approach using the inductive logic to learn the semantic rules for a query to deal with the ambiguities in it. One approach to map the natural language queries to SQL is to use Semantic Grammars. An example of such a recent work is PRECISE [9]. PRECISE is a system that guarantees the correct mapping of a natural language query to an SQL statement, if a query is semantically tractable. Different NLIDBs are constructed with thoughts and justifications. One of the examples is the construction of conversation based NLIDB for relational database. In this type of NLIDB, goal oriented or domain specific approach is followed to answer the queries using knowledge trees [10]. Knowledge trees help to keep the user's goal domain specific. Researchers have also worked on a concept known as NLKIDB [11], where K stands for knowledge base. This architecture uses the combination of NLIDB and Knowledge Base in order to map the queries to SQL more accurately when they are ambiguous. NLIDB portability is also a challenge and most of the NLIDBs designed so far are domain specific. In this regard, Marceleño [12] and his colleagues suggested ontology based flexible configuration of the domain for an NLIDB.

Some recent work has suggested NLIDB framework which uses ontologies to store the knowledge for a particular domain or database [13]. Also, these authors have introduced an intermediate language called DRS (Discourse Representation Structure) to transform a Natural Language Query into SQL [13]. Ontology can be very useful in improving NLIDB systems as it allows to abstract information and represents it explicitly highlighting the concepts and relations [14]. Vietnamese is an NLIDB system used by individuals and businesses to extract information from economic surveys. This system uses semantic grammars in order to map the parsed tree of a query [15].

Undoubtedly, natural language sentences can be ambiguous. There are several aspects of a sentence in natural language which need to be considered when processed by compilers or NLIDB engines. Pazos [16] presented a dialog based approach to deal with the semantic ellipses found during the process of parsing a query. Semantic ellipses refer to the missing information that adds full meaning to a sentence and helps to remove the ambiguities.

This work is about the identification and resolution of ambiguities while transforming a natural language query into SQL using AV Mapping Algorithm, which is an extension of our previous work [17]. The proposed algorithm is based on formal semantics and it efficiently maps a semantically tractable natural language query to an SQL statement. AV Mapping algorithm performs query transformation process by using a semantic dictionary. It reduces the number of comparisons significantly while mapping the value tokens to their relevant attribute/column tokens. However, natural language sentences are

ambiguous and in some cases an algorithm needs additional steps for the correct mapping of attributes and value tokens. The ambiguities which are found during the mapping process are divided into three categories and their resolutions are suggested in the context of AV Mapping Algorithm.

The main contribution of this paper is to incorporate improvements and enhancements in the AV Mapping Algorithm. The focus of improvement is on the identification of ambiguities and their resolution using AV Mapping Algorithm. The proposed improvements are totally novel and are specific to AV Mapping Algorithm. The structure of semantic dictionary is improved by introducing a new concept known as "Question Word Relevancy". This concept helps in mapping a question word found in the input query to a feature of an entity using a "Question Word Relevancy" table. The explanation of this concept is given in Section 5.3. AV Mapping Algorithm performs minimum processing while transforming a query from natural language into SQL as it follows the novel approach with the improved structure of semantic dictionary. For example, no parse tree is designed during the query transformation process, which is the unique feature of the proposed system. Instead, tokens are given an order number in the query scanning process, which is then used to map the query semantically using AV Mapping Algorithm. The same Semantic dictionary is utilized during the identification and resolution of ambiguities with the help of improved modules introduced in this work using AV Mapping Algorithm. This does not put any extra overheads on the processing module and keeps the algorithm efficient.

2. THE PROPOSED SYSTEM

2.1. Types of Questions

The natural language queries for Urdu language are divided in two categories: question or a request, as shown in Example-1 (request) and Example-2 (question). The words used for request or questions are the basis for the extraction of the required parameters from a query for its understanding. Furthermore, the positions of these words in a query have a vital role in identifying the types of parameters. These parameters are the name of a table, an attribute and the value of an attribute. As a whole, these three parameters in an SQL statement are required for the query to execute properly.

Example-1

مجھے امجد کی کلاس کا نام بتایں۔

[mʊʤeI] [ʌmjəd] [ki] [kɒlɒs] [kɒ] [nɒm] [bɒtɒyæñ][1]

[To me] [Amjad] [of] [class] [his/her] [name] [tell]

"Tell me Amjad's class name."

Example-2

امجد کی کلاس کونسی ہے؟

[ʌmjəd] [ki] [kɒlɒs] [kəʊnsi] [hæ]

[Amjad] [of] [class] [what/which] [is]

"What is Amjad's class name?"

2.2. Tokenization

To process a sentence, it is necessary to divide it first into chunks in order to understand its meaning and structure. In the process of tokenization, the sentence is divided in small chunks known as tokens. In an Urdu sentence, the words are separated by spaces as it is in English. The proposed system tokenizes a sentence into small pieces, which then undergoes further processing in other steps.

For simplicity and efficient transformation, each token category is given an order number. This ordering makes it almost effortless to identify the category of tokens for further processing. For example, if the processing of a token category is not required, that category is ignored for further processing.

2.3. The Role of Syntactic Markers

This NLIDB is based on formal semantics and deals with a natural language query semantically. It is necessary to define and ignore the syntactic markers for further processing as they do not have

[1] The special symbols used with each example of Urdu sentence are adopted from IPA chart. They represent phonemes of the Urdu characters. They are used when transliteration of Urdu words is required to facilitate non-Urdu speakers/readers to understand/read Urdu text. For more detail, visit http://www.cle.org.pk/clt09/download/Papers/Paper3.pdf.

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 make no semantic contribution to the interpretation of the question [9]. For example, a simple query in Urdu is shown in Example-3, below.

Example-3

اسلم کے والد کا کیا نام ہے؟

[ʌslɒm] [KeI] [vɒlId] [kɒ] [kIyɒ] [næm] [hæ]

[Aslam] [of] [father] [of] [what] [name] [is]

"What is Aslam's father's name?"

In this sentence, the word (کے) is a syntactic marker. It is being the part of a natural sentence and is concatenating the word (اسلم) to the word (والد), but when translating the sentence into SQL, the word "کے"does not contribute in the process of transformation to an SQL statement. In order to treat the natural query semantically, these syntactic markers are ignored. Syntactic markers merely help in the extraction of necessary parameters as they show the relationship between two words. In (3) above, the word "is" is connecting the question word "what" with the value token "Aslam" of required column "father's name". Syntactic markers are ignored in onward processing after extracting the required information with their help.

2.4. Extraction of Necessary Parameters

For successfully translating a natural language query into SQL, there is a need to identify the required parameters i.e. a table name, an attribute, and a value. To understand the extraction process of parameters, understanding the structure of the SQL is a pre-requisite. To start with, consider an SQL query (Q) given below:

Q = SELECT Name FROM Personal WHERE Name = 'Ahmad'

By looking at the query in the given example, the following conclusion can be drawn. In this Query "SELECT", "FROM" and "WHERE" are the operators (O), whereas "Name" and 'Ahmad' are the operands (OP). This structure implies the need to extract the operands and their respective operators from a natural language query. Consequently, this needs a lexicon/dictionary and a set of rules, which will be required to extract the needed semantic information from the input query. More details on the extraction of parameters are given in Section 3.

2.5. The Keywords

This section comprises the list of the keywords that are used by the system to identify the parameters and constructs. These make the job of the algorithm trouble-free in identifying the types of tokens in the input query. Moreover, the occurrence of these words in specific positions in a query helps the algorithm to fire an appropriate rule for the extraction of required parameters. In Table 1, the list of the keywords and their types are provided.

Table 1. Dictionary keywords.

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

2.6. Construction of the Dictionary

The aim of a natural language interface is to facilitate ordinary users of computer without long training. For this purpose, a domain specific dictionary is designed to keep the record of entities, their possible attributes and possible synonyms. The inclusion of synonyms makes it possible for the user to write a particular sentence in alternative ways. We call this dictionary as semantic dictionary, because it does not contain any syntactic information for tokens. It is used by Attribute Value Mapping Algorithm for the sake of query transformation. For attribute value mapping, there is a detailed discussion in Section 3.

Semantic dictionary contains the synonyms for words in each column and table in the database. It also contains the plural for each word because there is e.g. no addition of only "s" or "es" similar to English language, instead additional information is required in Urdu Plurals. In Urdu language, addition of specific character(s) at the end of a word is not consistent; therefore, plurals are being placed in the dictionary. Table 2 contains some sample words with their meanings and plurals, which gives a clear idea about the nature of plurals for different words.

Table 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 describes the structure of Urdu words and their plurals. As each word has a different plural, we do not need to define rules to convert a word into its plural. AV Mapping Algorithm uses this dictionary to identify the semantic patterns in the query, such as an entity, a table name, column, or an expected value for a column.

2.7. Semantic Knowledge Base

The aim of the constructed system is to ensure the correctness of a query semantically. For this, all the semantic information is made available in the semantic dictionary that is obligatory for the process of query transformation. An attribute value mapping algorithm is designed that efficiently transforms natural language query into its equivalent SQL using the semantic knowledge base. The semantic knowledge base contains three main tables, having semantic information for a database that is to be used by AV mapping algorithm. This algorithm is discussed in Section 3 in detail. Fig. 1 depicts the structure of the semantic knowledge base.

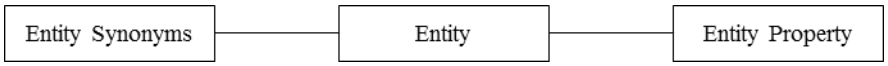

Fig. 1. Semantic knowledge base.

This dictionary is manually constructed and is domain specific. The dictionary is not like a huge corpus; rather it has entries according to the number of entities in the database. Use of this dictionary during query transformation makes the system efficient.

To understand 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 something about which an organization collects data [9, 7]. The steps that are essential for constructing this semantic dictionary are: (a) Identifying all the entities in a database, (b) Finding out and storing the synonyms of the identified entities, and (c) Defining the characteristics of these entities.

The storage of data in semantic dictionary with the help of an example is explained in Table 3. Here we suppose that, we are constructing the semantic dictionary for a small School Management System. For the sake of understanding, let us take only a few entities from the whole system. Suppose E denotes the entities, α, the synonyms and β, the properties of an entity E. We take three entities and construct the dictionary as shown in the Table 3(a-c).

Table 3(a). Database entities.

Entity	Meaning
طالب علم	Student
کورس	Course
جماعت	Class

Table 3(b). Entity synonyms.

Entities	**طالب علم**	**کورس**	**جماعت**
Synonyms	طالب		درجہ کلاس

Table 3(c). Entity properties.

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

From the given tables, we conclude that: (a) for each entity E_i there are a number of synonyms (α_i…….. α_n) so that E_i = (α_i…….. α_n). (b) Each entity has a number of properties (β_i…….β_n) so that E_i has (β_i…….β_n). Few sentences that highlight the deployment of these tables by the AV Mapping algorithm are presented below.

Example-4(a)	**Example-4(b)**	**Example-4(c)**
وسیم کہاں رہتا ہے؟	**پانچویں جماعت میں کتنی کرسیاں ہیں؟**	**کیمسڑی کے اجزاء کیا ہیں؟**
[W^SEIM] [kɒha:ñ] [rəhtɒ] [hæ]	[pɒntʃvñ] [dʒɒmɒt] [meIñ] [kItni] [kʊrsIyɒñ] [hæñ]	[kəmIstɒri] [keI] [ʌdʒzɒ] [kIyɒ] [hæñ]
[Waseem] [where] [live] [is]	[5th] [class] [in] [how many] [chairs] [are]	[Chemistry] [of] [contents] [what] [are]
"Where does Waseem live?"		
"Tell me Amjad's class name."	"How many seats are there in class 5?"	"What are the contents of Chemistry?"

In the above examples, one characteristic of each of the three entities is used as an evidence that how this semantic dictionary is utilized. In Example-4(a), a question has been asked about the Student entity in

the context of School Management System. Student has a property of living some whereas is given in Table 3(c) as (پتہ, address), 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 (وسیم, Waseem) is the name of a student. By using this concept, we do not look for the value token "Waseem" in the whole database and match it against every value of a possible column. Consequently, this idea makes the proposed system faster and more efficient in terms of mapping as it avoids the long and time consuming matching process.

In Example-4(b), the question has been asked about the number of chairs (کرسیاں) in class 5. The word (کرسی, chair) reflects that a user is asking the question about a class as this word is found in the column of class in Table 3(c). The Example-4(c) asks about the contents (اجزاء) of a course or subject. As the word "contents" is used in context of a course, which implies that the user is asking the question about the course. The algorithmic details are given in the next section on AV Mapping algorithm.

3. ATTRIBUTE VALUE (AV) MAPPING ALGORITHM

This section discusses the AV Mapping algorithm that efficiently maps a natural language query into its equivalent SQL using the semantic dictionary. The algorithm is divided into different modules. Each module is explained separately. Pseudo-Code Listing 1 shows the tokenization module of this algorithm.

Pseudo-Code Listing 1: Query tokenization.

```
//Tokenization algorithm
Module 1 (Scanning)
  - Start Scan
      1. Split the query (Q) in tokens (t1…tn)
      2. Give an order number to each token 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
          //Ignore tokens having ordert1
  -  End Scan
```

The algorithm starts its work by scanning the input query. Input query is tokenized on the basis of tokenization rules given in the algorithm. The identified tokens are then further processed for ordering/ranking. Ordering is done to simplify the job of the algorithm with the implication that only the required token categories are considered for further processing. Such token categories, which do not contribute any semantic information in terms of query interpretation for SQL are ignored. A key point in the proposed algorithm is that it does not consult the database to map a value token to its corresponding attribute/column token found in the input query. This feature minimizes the processing efforts to a great extent and makes the algorithm efficient, and differentiates the proposed system from other suggested systems. Also, AV Mapping algorithm assumes all unmarked tokens as value tokens that are left without any token mark. Furthermore, AV Mapping algorithm keeps both the attribute and value tokens with the same order number 3 that makes the algorithm more efficient to perform the mapping process for an attribute and value token on the same level. So, the algorithm performs the mapping process between the attribute and value tokens on the basis of relevancy rules. Relevancy rules work by looking at the positions of question words as well as required information words and try to map both by using semantic dictionary.

All the tokens which have order number 1 are ignored for further processing as they are either stop words or tokens that do not contribute to semantic information required for query transformation process. The step performed by the algorithm is the identification of the type of question. Questions are divided into two categories on the basis of their nature which is already discussed in section 2.1. If the question

lies in neither of the categories, the question is marked as intractable; otherwise it is forwarded for further processing. Another module of the AV Mapping Algorithm is given in Pseudo-Code Listing 2. In this module, the process of query break up and its storage in the form of attribute/value pattern is highlighted.

Pseudo-Code Listing 2: Pattern extraction.

```
//Algorithm that splits the query and extract patterns
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 of Ordered Formatted Query
```

Pseudo-Code Listing 2 shows the processing of the query splitter module. Once a query is scanned and unnecessary tokens are removed, it is broken into small chunks and each chunk is treated individually. The query is broken down on the basis of connector tokens or splitter/criteria tokens. In Urdu language (similar to 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), which respectively stand for "and" and "or".

The algorithm also checks the query for criteria tokens. If a criteria token is found, then the algorithm breaks up the query on the basis of these criteria tokens. If a query is semantically correct, there will be an attribute value pattern after criteria tokens. The examples of both the connectors and criteria tokens are shown in Example-5.

Example-5

سلمان کی تنخواہ کیا ہے اور وہ کہاں رہتا ہے ؟

[sʌlmɒn] [ki] [tʌnxʊːɒh] [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 basis of اور (and) token, it results in the following sentences.

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

Similarly, Example-6 shows the use of criteria words in a sentence and their break up through the splitter algorithm.

Example-6

مجھے ان طلباءکے نام بتایں جو پشاور میں رہتے ہیں-

[mʊʤheI] [ʊn] [tɒlbɒ] [keI] [nɒm] [bɒtaːyæñ] [ʤəʊ] [pIʃɒvə(r)] [meIñ] [rəhteI] [hæñ]

[to me] [those] [students] [of] [names] [tell] [who] [Peshawar] [in] [live those] [are]

"Tell me the names of those students who live in Peshawar?"

The criteria tokens are used to specify a condition in a query. In the query above, **جو** (who) is a criterion token. In Urdu language, the criteria token occurs between the required information and the attribute plus value. If the given query is broken on the basis of criteria token, the following chunks of query will be obtained.

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

After this splitting, the chunks will be forwarded to Module 3 for further processing, where AV Mapping will take place for every identified chunk of the query. The split of the query into chunks will provide an ease in identifying the relevant value tokens to their respective attribute tokens. 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 second chunk gives the address (رہتے, live in) and its respective value (پشاور), which is name of a city**.** Pseudo-Code Listing 3 describes the actual mapping in the AV Mapping Algorithm step by step.

Pseudo-Code Listing 3: Attribute value mapping.

```
//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 replaces 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
```

The AV Mapping algorithm shown above efficiently maps assumed value tokens to their respective attribute tokens. If the query is not in proper format, the value tokens fail to map to their respective attribute tokens. If a failure occurs during the process of mapping a value token to its respective attribute token, the query is marked as intractable. By contrast, if the query lies in one of the categories as either a proper question or request, the values are successfully mapped and we get an intermediate form of the query which is effortlessly transformable into SQL.

4. AMBIGUITY IDENTIFICATION AND RESOLUTION

Ambiguity refers to a situation when a single sentence has more than one meaning. In this research, there are three kinds of ambiguities that are identified during the transformation process using AV Mapping Algorithm. The ambiguities are listed as: (a) Disqualified reference to an Entity, (b) Multiple attributes competence and (c) Ambiguity due to insufficient information.

4.1 Disqualified Reference to an Entity

This ambiguity was found because of reference of an entity during the query transformation process. In this type of ambiguity, an entity occurs which is not clearly qualified. The entity is called "disqualified" because it occurs in the form of a value and does not explicitly contain the information as to which table it belongs. In other words, the value qualifies to be called an anonymous entity.

Example-7

مجھے جمال کی عمر بتایں؟

Tell me the age of Jamal?

[mʊʤeI] [ʤɒMɒL] [ki] [OʊMɜ˞] [bɒtɒyæñ]

[tome] [Jamal] [of] [age] [Tell]

In query given in Example-7, "Jamal" is a person's name and is referring to an entity which is not provided in the query. The only thing which is provided is age, which is both the attribute of an employee and student entity. During the process of mapping, AV Mapping Algorithm finds the two matching entities which are a "student" and an "employee". So, here an ambiguity arises for the property referral. However, this ambiguity only occurs when two or more entities have the same attribute or property. See example 8 to explain this situation.

Example-8

اسلم کا پتہ کیا ہے؟

What is the address of Aslam?

[ʌslɒm] [kɒ] [Pɒtɒ] [kIyɒ] [hæ]

[Aslam] [of] [address] [What] [is]

Example-8 is a query that is asking about the address of a person whose name is "Aslam". This is not an ambiguous query if there is only one entity in the database that has address property. But in a database such as LMS (Learning Management System) there are both students and teachers which are clearly separate entities and have "address" as a common attribute. Even sometimes the values of some attributes can be the same. Making all this as a base the given situation is termed as ambiguous.

4.2 Multiple Attributes Competition

This kind of ambiguity is observed by the AV Mapping Algorithm when it finds multiple attributes competing for a single value. There are multiple ways of asking the same query in natural language. In Example-9, a sample query is given where ambiguous situation is identified.

Example-9

اسلم کے والد کا کیا نام ہے؟

What is the name of Aslam's father?

[ʌslɒm] [keI] [vɒlId] [kɒ] [kIyɒ] [næm] [hæ]

[Aslam] [of] [father] [of] [What] [name] [is]

In the query above, a question has been asked about the male parent of a person named as "Aslam". During the process of transformation, AV Mapping Algorithm finds two columns "name" and "father" and a single value "Aslam". Actually, the user is asking about Aslam's father's name. During the process of matching properties using semantic dictionary, the algorithm finds "father" and "name" as two separate columns. Both are the properties of a person. This is an ambiguous situation for AV Mapping Algorithm and needs to be resolved, which is fully discussed in Section 5. Now, consider an alternate of Example-9, which does not create an ambiguous situation if the combination and selection of words are changed. See Example-10 for the alternate query.

Example-10

اسلم کا والد کون ہے؟

Who is Aslam's father?

[ʌslɒm] [kɒ] [vɒlId] [KAʊN] [hæ]

[Aslam] ['s] [father] [Who] [is]

In this example, the user is asking about Aslam's father's name, which has the same meaning as of Example-10, but with different combination of words. In this query, there is just one attribute/property given, which is father. By using Semantic dictionary, this mapping can be achieved without any ambiguity as father is the property of a person.

4.3 Ambiguity Due to Insufficient Information

This kind of ambiguity is found in queries that have insufficient information required for mapping by AV Mapping Algorithm. Such queries cannot be processed further by the AV Mapping Algorithm as there are no attribute/property tokens in the query which are necessary for mapping. For example, there can be a query that has no column but only a value. Example-11 and Example-12 are given that have such ambiguities.

Example-11

سلمی کون ہے؟

(a) Who is Salma?

[SɒLMɒ] [KAʊN] [hæ]

[Salma] [Who] [is]

Example-12

پشاور کہاں ہے؟

(b) Where is Peshawar?

[pIʃɒvə(r)] [kɒhɒñ] [hæ]

[Peshawar] [Where] [is]

In Table 4, the tokens and their respective categories are shown both for sentence (a) and (b) given in Example-11 and Example-12. If both the queries are observed in these examples, they do not contain any attribute tokens but have only value tokens. By looking at the working of AV Mapping Algorithm, it is necessary to have an attribute for a value to be mapped using Semantic Dictionary and some intelligence. Table 4 depicts the situation of identified tokens in both the sentences.

Table 4. Tokens assignment.

Tokens Sentence (a)	Tokens Sentence (b)	Token Category
Who	Where	Question word
is	is	Stop word
Salma	Peshawar	Value token

5. RESOLUTION OF IDENTIFIED AMBIGUITIES

In order to make the suggested solution robust, all the identified ambiguities should be resolved in an acceptable way from the user's point of view. There are three kinds of ambiguities that are identified in this research. For each category, the solution is given below in detail.

5.1 Resolving Disqualified Reference to an Entity

For such kind of ambiguity to occur, it is assumed that a property has more than one entity matches during the process of attribute value mapping using AV Mapping Algorithm. There are two approaches suggested to resolve this ambiguity. The first solution suggests the following steps:

i) Consider the statistics of most queried entity and map the query with respect to that entity straightaway

ii) For convenience, also show other matching entities if the first one is not the desired answer for the user

The second approach suggests a simple and straight-forward procedure. In this case, before mapping the attribute to an entity, the user is given with the option to select the desired entity among all the matched entities. The user selects one entity and the query is mapped with respect to that entity. The improved version of this part of the AV Mapping Algorithm is given below in Pseudo-Code Listing 4.

Pseudo-Code Listing 4: Improved version of algorithm for resolving ambiguity mentioned in 5.1.

```
//Improved Version of Steps for Resolving Ambiguity 5.1
Module 3 AV Mapping (queryparts)
-Start Making AV Patterns
For each chunk ci=1 to ∑c in queryparts
   //All the previous steps repeated + Resolution first option
   If (ci count(attr) > 0 &&attr has count(Entities) > 1)
     i-   Pick up all the matching entities entities for attr
     ii-  Select one having greater stats value
     iii- Keep other in the list to show as an option
   End If
   //Resolution with second option
   If (ci has att&&attr has count(Entities) > 1)
     i-    Pick up all the matching entities entities for attr and prompt
           user to select the one as per user choice
     ii-   Map the answer as per user selection
   End If
-End AVPatterns
```

5.2 Resolving Multiple Attributes Competition

In this case, multiple attributes ambiguity occurs in a query, whereas in Section 5.1 a single attribute was matched with multiple entities. This kind of ambiguity is resolved using the token relevancy rule. This rule suggests that if there are multiple attributes against a single value, check the relevancy of each attribute with the value. The result of this matching will give a better idea of understanding the relationship between attributes. If both the attributes are referring to the same entity, then take a single attribute and map it for query transformation. For example, in Example 10 above the relevancy token is apostrophe ('s), which connects the attributes "name" and "father". Its examples in Urdu are (کا، کے), which work in the same way as an apostrophe works in English and connects the two relevant attributes. The improved version of the relevant module of the algorithm is given in Pseudo-Code Listing 5.

Pseudo-Code Listing 5: Improved version of algorithm for resolving ambiguity mentioned in 5.2.

```
//Improved Version of Steps for Resolving Ambiguity 5.2
Module 3 AV Mapping(queryparts)
-Start Making AV Patterns
For each chunk ci=1 to ∑c in queryparts
  //All the previous steps repeated + Resolution of Ambiguity 5.2
   If (count(attr) > 1 && count(val) == 1)
      i-    Check for apostrophe token and take the adjacent_attr
      ii-   Match other attr in SD for the same entity
      If(Ref_Entity(attr1) == Ref_Entity(attr2))
      AvPatterns[counter] = adjacent_attr
      End If
   End If
 End For
-End AVPatterns
```

5.3 Resolving Ambiguity with Less Information

This kind of ambiguity occurs when the information provided in the query is less than the required information by AV Mapping Algorithm. In order to deal with this kind of ambiguity, the structure of the Semantic Dictionary is upgraded and an additional characteristic for an entity is added. This column is given the name "Question Word Relevancy". This column keeps the possible question word for an entity and on the basis of this word the entities are selected and presented to the user for confirmation. The user then confirms his selection with the presented choices. Table 5 elaborates the improved structure of the dictionary.

Table 5. Concept of question word relevancy.

Entity	Question Word Relevancy
Student (Person)	Who
City	Where

The proposed solution to this kind of ambiguity behaves in an intelligent manner. It makes the AV Mapping Algorithm more intelligent when a query is found with incomplete information required for the transformation process. In Table 5, two sample entities and their relevant question relevancy words are shown. In order to utilize this feature, whenever AV Mapping Algorithm finds a query with insufficient information, it will call another module to look at the question word relevancy category. Consider the following example:

Example-13

Who is Adnan?

عدنان کون ہے؟

[ʌdnɒn] [KAʊN] [hæ]

[Adnan] [Who] [is]

Example-13 shows the query with insufficient information. In the query, someone is asking about (Adnan, عدنان), which is a person's name. In other words, the query contains only value token without any mapping attribute that helps to map the value to a right entity. To deal with such situation, AV Mapping Algorithm will call its "Question Word Relevancy" finder module and will pick up all the entities related to the relevancy class. In Example-13, the relevancy class is (Who, کون), which implies that "Adnan" is a student or any person. Consequently, the query will map the value to the entity picked by the question relevancy word. Consider another example:

Example-14
Where is Peshawar?
پشاور کہاں ہے؟
[pIʃɒvə(r)] [kɒhɒñ] [hæ]
[Peshawar] [Where] [is]

In Example-14, a user asks about the location (Peshawar, **پشاور**), which is the name of a city. According to AV Mapping Algorithm, the only semantic information in the given query is "Peshawar", which is a value token. This is insufficient information for the algorithm to map the query into SQL. Hence, the algorithm consults the "Question Word Relevancy" module. The module finds and lists the matching entities from the semantic dictionary for the question word "Where". According to Table 5, the matching entity is "City", which in this case will be mapped to "Peshawar" as its relevant attribute. Pseudo-Code Listing 6 describes the portion of algorithm used for the resolution of this type of ambiguity.

Pseudo-Code Listing 6: Improved version of algorithm for resolving ambiguity.

```
//Improved Version of Steps for Resolving Ambiguity 5.3
Module 3 AV Mapping(queryparts)
-Start Making AV Patterns
 For each chunk ci=1 to ∑c in queryparts
   //All the previous steps repeated + Resolution of Ambiguity 5.2
   If (count(attr) == 0 && count(val) == 1)
      i-    Look for the relevancy question word against an entity
      ii-   If matched pick the default col to map it to value
 End If
```

6. QUERY TRANSFORMATION

After a query is processed by the AV Mapping algorithm, all the necessary information is extracted that is required transforming the query into SQL. Fig. 2 shows all the steps which are performed in the process of query transformation.

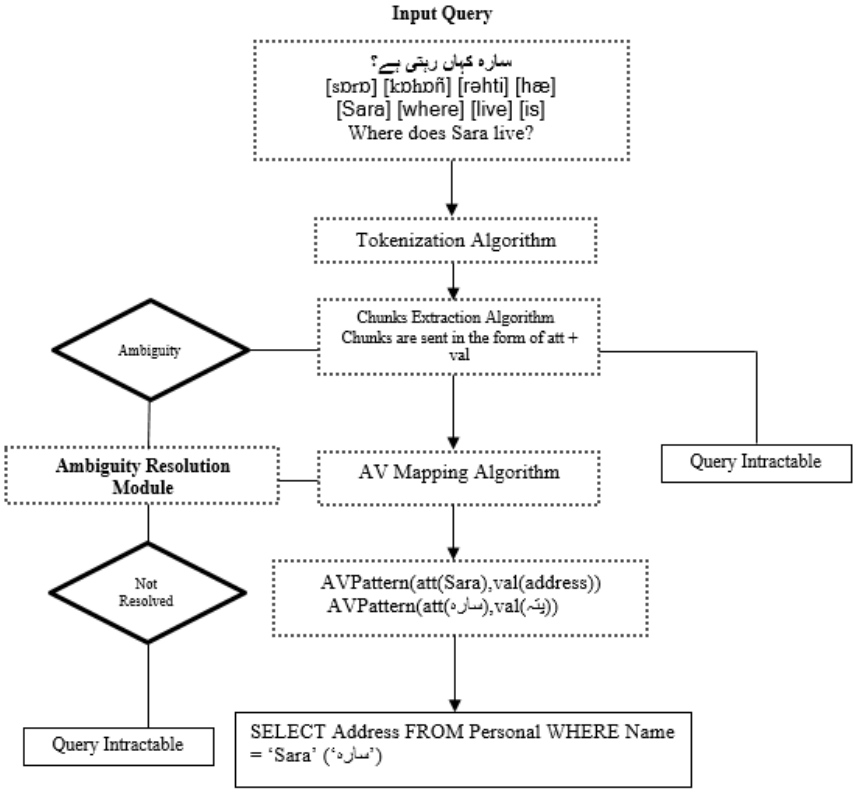

Fig. 2: Query transformation.

In query transformation, a natural language input query is tokenized by the tokenization algorithm. The tokens bear only semantic tags and no part of speech tags. As the approach is all about semantic grammars, therefore, all the syntactic markers are ignored and only the required tokens are selected, paired and are ready to be forwarded to the AV Mapping Module of the Algorithm. However, some rules for ambiguity occurrence are checked before these tokens are forwarded. If ambiguity exists, the chunks are forwarded to the ambiguity resolution module. If the input query is not in the required format, it is considered as intractable query.

7. EXPERIMENTAL RESULTS

The algorithm was tested on two query sets which were collected from the relevant users of the selected domains. One query set was for School Management System and another was for Employee Information System. From the selected domain users, queries were taken on the paper. Users were guided about the nature of the expected queries. Testing was kept limited to queries which are frequently used in one of these two systems on daily basis. The queries which are complex and are used in decision making were avoided. For the selected set of queries, results were up to the mark as AV Mapping Algorithm mapped most of the queries. A summary of the experimental results of the tests is shown in Table 6.

Table 6. Experimental results.

DB Name	Total Queries	Statistics					Overall Accuracy Percentage		
		QT	Total	Correct	Incorrect	Accuracy %			
SMS	200	ST	95	89	6	93	Correct	Incorrect	Accuracy %
		CMP	51	44	7	86	176	24	88
		AMB	43	35	8	81			
		INT	11	8	3	73			
		QT	Total	Correct	Incorrect	Accuracy %			
EIS	200	ST	82	76	6	93	Correct	Incorrect	Accuracy %
		CMP	59	48	11	81	169	31	84.5
		AMB	47	38	9	81			
		INT	12	7	5	58			

SMS: School Management System **CMP:** Complex
EIS: Employee Information System **INT:** Intractable (Queries with Less Info)
ST: Straightforward

The queries considered for test were simple and straight forward. The users of the relevant domains were guided about the nature of the expected queries before the queries were taken. Most of these queries required the data from two or three tables. The query sets are categorized into three categories (i) Straightforward Queries, (ii) Complex or Ambiguous Queries and (iii) Intractable Queries, as described below.

Straightforward queries are those queries that contain enough information required for the AV Mapping Algorithm to map it into SQL. In this category, AV Mapping algorithm can extract the required information in its default way without any additional treatments i.e. the attribute value patterns it identifies are clearly connected and can be mapped to each other using semantic dictionary. As the subject-verb-object pattern is similar in both English and Urdu queries, therefore, English queries are

explained in the following Examples. In order to understand the nature of such queries consider Examples 15-18.

Example-15

What is Salma's class?

سلمی کی کلاس کونسی ہے؟

[SɒLMɒ] [ki][kɒlɒs] [kəʊnsi] [hæ]

[Salama] ['s] [class] [What] [is]

In the given example, "class" is found as an attribute of a student entity while (Salma, **سلمی**) is a value. Using semantic dictionary, these tokens are verified for entity characteristic relevancy. If relevancy is found valid, these tokens are mapped and forwarded for query transformation.

Ambiguous queries are those that have something missing and need an additional step in the algorithm to be treated properly. The queries in this category contain the required information, needed by AV Mapping Algorithm, to transform it into SQL but the identified attribute has the tendency for multiple entities. In order to understand the nature of such queries, consider Example-16.

Example-16

How many marks chemistry has?

کیمسٹری کے کتنے نمبر ہیں؟

[kəmIstɒri] [keI][kItnɒ] [NʌMBɜ˞] [hæñ]

[Chemistry] [of] [How many] [marks] [has]

In the query above, (marks, **نمبر**) is an attribute and "Chemistry" is its mapping value for subject. "Subject" is an entity in student information system and "marks" is a valid property for subject. Similarly, in the same system "Student" is an entity and "marks" is its valid property. While matching the attributes in semantic dictionary using AV Mapping algorithm, the property is matched to both the entities, which becomes ambiguous for the algorithm in the mapping process. The solution for this situation is already given in the Section 5.2.

The third query set is titled as incomplete queries. This set of queries contains insufficient information for the AV Mapping Algorithm to map it into SQL. That is, the query in this case contains a value without any mapping attribute, which creates ambiguity. In order to deal with such queries, the question token is extracted and its relevancy is checked against the relevant entity in the semantic dictionary. Handling such a query is given in Example-17 below.

Example-17

Who is Aslam?

اسلم کون ہے؟

[ʌslɒm] [KAʊN] [hæ]

[Aslam] [Who] [is]

In this example, (Aslam, **اسلم**) is a value without mentioning any of its relevant attribute. The only thing which helps the AV Algorithm to map such situation is question word, which in this case is "Who". The details of its treatment are already mentioned in Section 5.3.

The queries which are not mapped correctly by AV Mapping Algorithm are implicit summaries (such as queries asking for aggregate results as shown in Example-18 and queries containing incomplete information as shown in Example-19). Ambiguous situations identified in the paper are resolved correctly in most of the cases. However, a few situations that require additional intelligence level cannot be handled at present.

Example-18

How many working hours Amjad has this week?

وسیم نے اس ہفتے کتنے گھنٹے کام کیا ہے؟

[W^SEIM] [neI] [Is] [HɒFteI] [kItneI] [Ghɒntɒeɪ] [kɒm] [kIɒ] [hæ]

[Waseem] [has] [this] [week] [How many] [hours] [work] [done] [is]

Example-19

Tell me Ahmad's marks in Mid Term.

مجھے مڈٹرم میں احمد کے نمبر بتایں-

[mʊʤeI] [MIDTɜ˞M] [meñ] [ʌhmɒÐ] [keI] [NʌMBɜ˞] [bɒtɒyæñ]

[To me] [Mid Term] [in] [Ahmad] [of] [marks] [Tell]

8. CONCLUSION AND FUTURE WORK

In this paper, an AV Mapping Algorithm is presented that accurately maps natural language queries into SQL with minimum transformation time. The concept of semantic dictionary is proposed for the identification and resolution of ambiguities during complex query transformation, which increases the accuracy of mapping. Three types of queries are identified and solutions are devised for their accurate mappings. The proposed algorithms are generic and can equally likely be used for other similar languages as well.

Intelligent learning module can be incorporated in the AV Mapping Algorithm to make it more user-friendly. This module will enable interaction with the software in an easier manner. For example, it may provide the users with auto complete feature while writing the query. For this purpose, a knowledge base is required with accurate rules. Similarly, automatic creation of the semantic dictionary is scheduled in an easier way.

9. REFERENCES

1. Kumar, A. & K.S. Vaisla. Natural language interface to databases: Development techniques. *Elixir Computer Science and Engineering* 58: 14724-14727 (2013).
2. Knowles, S. A natural language database interface for SQL-tutor. *Honours Project Report,* p. 1-35 (1999).
3. Lopez, V., E. Motta, V. Uren, & M. Sabou. *State of the Art on Semantic Question Answering: A Literature Review*. Technical Report kmi-07-03. Knowledge Media Institute, The Open University, Milton Keynes, UK (2007).
4. Rao, G., C. Agarwal, S. Chaudhry, N. Kulkarni, & D.S. Patil. Natural language query processing using semantic grammar. *International Journal on Computer Science and Engineering* 2(2): 219-23 (2010).
5. Martin, P., D.E. Appelt, B.J. Grosz, & F. Pereira. TEAM: an experimental transportable natural-language interface. In: *Proceedings of 1986 ACM Fall Joint Computer Conference*, IEEE Computer Society Press, p. 260-267 (1986).
6. Minock, M. *STEP: A Natural Language Interface to Database*. Available: http://www.cs.umu.se/~mjm/. (Retrieved: February 15, 2016).
7. Sultan, A.A. *Natural Language Interfaces*. M.Sc Thesis. Department of Computer Science, University of Peshawar, Pakistan (1993).
8. Tang, L.R. Using a machine learning approach for building natural language interfaces for databases: application of advanced techniques in inductive logic programming. *Journal of Computer Science, Informatics & Electrical Engineering* 2(1): 140-60 (2008).
9. Popescu, A. Maria., O. Etzioni, & H. Kautz. Towards a theory of natural language interfaces to databases. In: *Proceedings of the 8th International Conference on Intelligent User Interfaces.* ACM, p. 149-157 (2003).
10. Owda, M., Z. Bandar, & K. Crockett. Conversation-based natural language interface to relational databases. In: *International Conferences on Web Intelligence and Intelligent Agent Technology – Workshops (IEEE/WIC/ACM)*, IEEE, p. 363-367 (2007).

11. Axita, S., P. Jyoti, P. Hemal, & P. Namrata. NLKBIDB-Natural language and keyword based interface to database. In: *International Conference on Advances in Computing, Communications and Informatics (ICACCI),* IEEE, p. 1569-1576 (2013).
12. Marceleño, J.A.Z., A. Gelbukh, & R.A. Pazos. Customization of natural language interfaces to databases: beyond domain portability. In: *Proceedings of Mexican International Conference on Computer Science*, IEEE Computer Society, p. 373-378 (2009).
13. Li, H., & Y. Shi. A WordNet-based natural language interface to relational databases. In: *Proceedings of 2nd International Conference on Computer and Automation Engineering (ICCAE),* IEEE, Vol. 1, p. 514-518 (2010).
14. Rajeh, K., & S. Safwan. Toward enhanced natural language processing to databases: building a specific domain ontology derived from database conceptual model. In: *7th International Conference on Informatics and Systems (INFOS)*, IEEE, p. 1-8 (2010).
15. Dat, T., D. Tam, & B. Son. Vietnamese natural language interface to database. In: *Proceedings IEEE Sixth International Conference on Semantic Computing*, IEEE, p. 130-133 (2012).
16. Pazos R.A., P.J. Rojas, S.R. Santaolaya, F.J. Martinez, & B.J. Gonzalez. Dialogue manager for a NLIDB for solving the semantic ellipses problem in query formulation. In: *Proceedings International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, Part II. Springer, Berlin,. p. 203-213 (2010).
17. Ahmad, R., M.A. Khan, & R. Ali. Efficient transformation of a natural language query to SQL for Urdu. In: *Proceedings of the Conference on Language & Technology (CLT09)*, p. 53-59 (2009).