Modeling Query Events in Spoken Natural Language for Human-Database Interaction

Omar U. Florez and SeungJin Lim
Computer Science Department
Utah Sate University
Logan, UT 84322-4205, USA

1. Introduction

The database-related technologies have been extensively developed over the past decades and are used widely in modern society. In response to the increasing demands upon high performing database interaction, many efforts have been made to improve database system performance. For example, indexing technologies enable us to efficiently retrieve information from very large databases [1]. However, the user interfaces to database systems essentially remain unchanged: SQL is the *de facto* standard language used either directly or indirectly through an API layer. It is, however, interesting to note that the most common mode of human interface is the communication through a natural language, which motivates us to use a natural language as a human-database interface [2]. Let us compare SQL and natural language in a database query.

– While a query in natural language expresses the mental representation of the goal by the user, an SQL expression describes the structure of the data stored in the database [3].

– While SQL is a declarative language to describe a fixed set of actions to manipulate data in a relational database, the set of actions supported by the verbs in a natural language [4] is large and can be extended.

– Nevertheless, the primary user intention of data manipulation is the same both in SQL and natural language.

Clearly, there is an important gap to be filled between the cognitive model of the user interaction with databases by humans in natural language and the structured model in SQL. This gap represents the different layers of the abstraction of the user interaction carried out by the user and by the database system over the data. In this work, we attempt to bridge the gap between spoken natural language and SQL to enable the user to interact with the database through a voice interface. This goal is achieved by recognizing a *query event*, whose structural complexity is moderate, presented in a spoken natural language phrase, and then translating it to an SQL query. The presence of a query event in a spoken language expression is detected in our approach by recognizing linguistic patterns in which a verb triggers a query action followed by a set of words which specify the action. In other words, our approach works as a linguistic *adapter* which converts the structure of the spoken language expression to that of SQL at three different abstraction layers: lexical, syntactic,

and semantic. The structure to be identified and the corresponding actions involved in these three layers are summarized in Table 1. They will be discussed in depth in later sections.

Abstraction layer	Spoken natural language	Actions to bridge natural language and SQL	SQL
Lexical	Voice signal from the user	Detection of phonemes from the voice signal and word formation from the phonemes	Words
Syntactic	Parts of speech of words by the proposed grammar	Syntactic analysis of parts of speech	Keywords and literals by the SQL grammar
Semantic	Queries in natural language	Representation of queries as events in a word stream	No meta data or events are considered. Syntactically valid actions are to be executed.

Table 1: The characteristics of the three linguistic abstraction layers in spoken natural language and SQL.

There are a wide range of interesting applications that can benefit from the proposed spoken language-based database interaction, ranging from an alternative user interface to perform database queries to voice-enabled retrieval of medical images in surgery. To this end, our contribution to the study of human-computer interaction is threefold. First, we formalize the detection of the presence of database queries in the user speech by modeling them as special events over time, called *query events*. Second, we provide a mechanism to translate a query event in a spoken natural language to an SQL query. Finally, we developed an application prototype, called Voice2SQL, to demonstrate the proposed user-database interaction approach in an application.

2. Previous Work

The work to enhance the interaction between humans and database systems has evolved over time. The approaches that are found in the literature can be classified into two by the way in which the interaction is carried on: textual and visual. While the use interaction in textual query systems (TQS) consists of typing SQL sentences using a keyboard, in visual

query systems (VQS) the human interaction is assisted by the visual representation of the database schema by means of a graph which includes classes, associations and attributes [5, 6]. In VQS, the user formulates the query by means of a direct manipulation of the graph. The output of the query is also visualized as a graph of instances and constants. One advantage of this approach is that users with limited technical skills still can access the database without too much effort. In recent years, Rontu *et al*. provided an interactive VQS on general databases and Aversano *et al*. suggested in [7] an alternative visual approach that employs an iconic query system in the interaction with databases in which each icon is a semantic representation of attributes, relationships and data structures. In contrast to other VQSs, this iconic query system provides a high level language that expresses queries as a group of different icons. Moreover, the output of a query is also an icon which can be reused in further operations. Catarsi and Santucci made a comparison between TQS and VQS in [8] and concluded that visual query languages are easier to understand than traditional SQL expressions. While all the previous works have been proved to be effective for human interaction with databases and are getting momentum in recent years, a query system exploiting spoken natural language is rare in the literature. In fact, a spoken query system (SQS) may have unique advantages as summarized as follows:

– The use of VQS and TQS may be restricted in some scenarios. For example, visually impaired people or people with Parkinson's disease may have hard time to use a mouse or a keyboard in such a way these devices are primarily designed. The voice can be an alternative interaction method to databases and will allow us to set aside visual representations and pointer devices [9]. Thus, an SQS may provide a general purpose interface which only relies on the user's voice captured through a microphone.

– From the ergonomics point of view, the use of concurrent and similar input methods increases the user's mental load and produces interference during the interaction with computers [10]. Hence, the degree of interference in the interaction can be reduced when we use an alternative interaction method like a spoken interface. As an example, a surgeon might need to retrieve medical images of the patient while he is operating. However, the use of traditional methods to retrieve images (e.g. pointer devices or keyboards) may distract the surgeon's attention which is focused on his hands and sight. A voice-enabled interface seems to be a suitable alternative interface in this context.

– The formulation of queries in natural language is generally more intuitive for users than the use of text-based queries. Li *et al*. showed in [2] that a natural language query interface leads to the creation of queries with a better quality than a keywordbased approach. The quality in queries was measured in this study in terms of average precision and recall of different queries.

3. A Linguistic Approach from Human Voice Queries to SQL

The Merriam-Webster dictionary defines a language as "a formal system of signs and symbols including rules for the formation and transformation of admissible expressions" [11]. This definition stresses the presence of a well-defined set of symbols and rules in a language. Both natural language and SQL can be thought of as two languages with different layers of abstraction of the underlying user intention; natural language being a higher abstraction than SQL. Since natural language is used in everybody's daily life, it would be an appealing alternative interface for interaction with computers to most users as long as it

can be understood by computers. However, the potential ambiguity in the meaning of a natural language expression remains to be resolved. In fact, the translation of sentences from natural language to SQL is still far from a satisfactory solution.
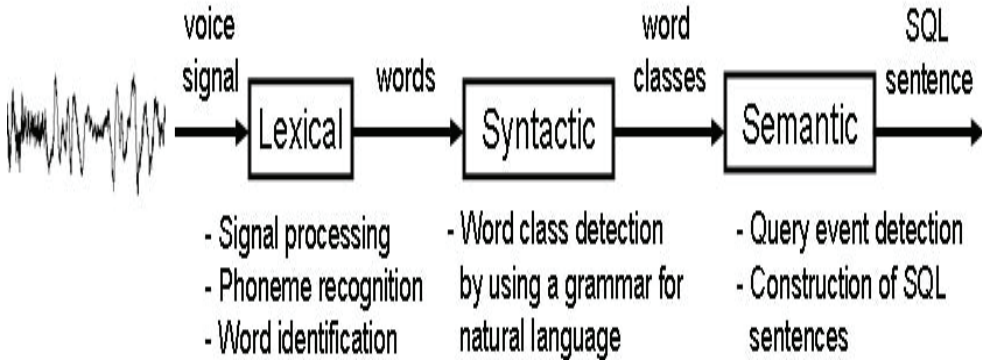

Fig. 1. Flow of data through the three processing components involved in the translation of queries in spoken natural language to SQL queries.

Our approach to the translation of query phrases from a spoken natural language to SQL is mainly to detect *query events* (which are discussed later) by three processing components, *lexical*, *syntactic* and *semantic*, as depicted in Figure 1, and then translate them as an SQL expression. The lexical component models common low-layer symbols in both languages to generate words. The syntactic component exploits the arrangement in the words to distinguish the associated word classes, e.g., adjective, verb, determiner, subject, preposition, and adverb, by employing a context-free grammar for natural language. Once the class of each word is identified, the semantic component checks for valid query events and builds SQL sentences.

Example 1. Consider the next sentence in Spanish[1] as an example of a query pronounced by the user.

Recuperar el nombre, el curso y la nota de los alumnos que tengan un profesor
el cual les enseña un curso.

which is translated in English as follows:

Retrieve the name, the course and the grade of the students that have a lecturer
which teach them a course.

We will use this sentence as an aid in presenting the proposed approach in this chapter. !

3.1 Lexical Component

The lexical component of our approach represents the lowest level of abstraction of the input voice signal captured via a microphone. While the words pronounced by the user are the basic units of information in spoken natural language, SQL employs a set of keywords and user-defined literals to express a query expression. Our goal here is to recognize words expressed in the input voice signal. To this end, we apply a typical signal processing technique that involves splitting the input signal into small segments by considering significant pauses in the signal as delimiter, digitizing each signal segment into discrete values, applying a machine learning algorithm to the discrete values to detect phonemes

from them, and recognizing valid dictionary words. We believe that the detection of silent periods in the voice signal and subsequently phonemes, such as /r/, /e/, /c/, /u/, /p/, /e/, /r/, /a/, and /r/ in *recuperar* (*retrieve* in English) leads us to the recognition of the words pronounced by the user. Since phonemes have a short duration, we analyze each signal segment by using short-time slicing windows of time length w with the goal of finding phonemes inside. The window length $|w|$ is an external parameter and should be long enough to contain phonemes. In the literature this value is commonly set to a value between 15 to 25 milliseconds [12, 13], and set to 20 milliseconds in our approach. Since some phonemes may not be detected if they appear in the two consecutive time windows $w1$ and $w2$, we let the third time window $w3$ overlap $w1$ and $w2$ half way in order to capture the eventual presence of those phonemes. From each window slice, we extract the most representative features as a vector for further processing. This procedure, denoted as *segmentation* in the literature [14], is repeated for the entire voice signal as it arrives through the microphone.

Among the number of descriptors to extract features from a voice signal within a fixed length time window, such as Linear Predictive Coding [15], Perceptual Linear Predictive [16] and RASTA [17], Mel-Frequency Cepstrum Coefficients (MFCC) are widely used because they have shown to be a robust and accurate approximation method [18]. In practice, a feature vector of 12MFCC coefficients is enough to characterize any voice segment. In other words, the entire spoken query can be divided into segments and each segment is characterized by a feature vector of 12 coefficients. Clustering of these vectors helps us identify the phonemes contained in the input signal. Among the several existing clustering methods, we chose the Kohonen's Self-Organizing Map (SOM) which is trained with a set of feature vectors, each of which is labeled with a phoneme. The detection of phonemes in the user speech is then reduced to obtain the neuron with the most similar feature vector on the SOM. The shape and members of the clusters on the SOM map changes over time while the SOM learns different phonemes through successive iterations. After training the SOM, we perform a calibration step where a set of feature vectors with well-distinguished labels are compared against the map. An example of the resulting map after training and calibration is depicted in Figure 2. (The signal processing details and program parameters used in our lexical procedure are fully explained in [19].)

Example 2. Consider the input spoken query in Example 1 again. The SOM training after calibration recognizes phonemes from the corresponding voice signal. Examples of the detected phonemes include: /r/, /e/, /c/, /u/, /p/, /e/, /r/, /a/, and /r/ for "recuperar" (*retrieve*), /l/, /a/ for "la" (*the*) and /n/, /o/, /m/, /b/, /r/, and /e/ for "nombre" (*name*). Once phonemes are recognized from each signal segment, the detection of words becomes our final task at this layer. It seems reasonable to think that a sequence of phonemes forms a word, but some words may not be correctly formed since the presence of noise in the feature extraction process may lead to the recognition of false positive or false negative phonemes. Since we are interested in obtaining dictionaryvalid words only, we approximate each word, formed by a sequence of phonemes, to the most similar word in a dictionary by using the *edit distance* as similarity function.

3.2 Syntactic Component

We have obtained a sequence of valid words from the previous lexical component. In the syntactic component, we employ a lightweight grammar to discover the syntactical

Fig. 2: Illustration of the SOM after the training and calibration process. Note that some neurons have learned certain phonemes. For example, the phonemes 'ñ', 'y' and 'z' are treated differently from others in Spanish due to their unique pronounciation. These phonemes are located, individually or together, in clusters isolated from other phonemes.

class of each word such as noun, verb, determiner, and adjective from the given word sequence.

There are different types of grammars that define a language such as context-free, context-sensitive, deterministic, and non-deterministic. Although all natural languages can be easily represented by context-sensitive grammars which enable simpler production rules than other types of grammar, the problem of detecting if a language is generated by a context-sensitive grammar is *PSPACE-complete* [20], making it impractical to program such a language. In contrast to context-sensitive grammars, context-free and deterministic grammars (Type-2 in the Chomsky hierarchy) are feasible in programming without generating ambiguous languages and can be recognized in linear time by a finite state machine. Thus, we propose a context-free, deterministic grammar to process the word sequence. The proposed grammar is shown in Figure 3 by means of the Backus-Naur Form notation. The production rules in the grammar identify the syntactical class of each word.

Example 3. By applying the proposed grammar to the words found in the user query sentence, we obtain the class of each word as follows: *recuperar (retrieve:verb), el (the:determiner), nombre (name:noun), el (the:determiner), curso (course:noun), y (and:copulative-conjunction), la (the:article), nota (grade:noun), de (of:preposition), los (the:determiner), estudiantes (students:noun), que (that:preposition), tienen (have:verb), un (a:determiner), profesor (lecturer:noun), el (the:determiner), cual (which:preposition), les (them:determiner), enseña (teach:verb), un (a:determiner), curso (course:noun).* !

3.3 Semantic Component

In our work, user queries given in the voice stream data are recognized as especial query events from the sequence of (word:class) pairs generated by the syntactic component. For this purpose, we propose an event model as follows:

Definition 1 (user query event) A user query is an event that consists of five event attributes

<What, Where, Who, When, Why>

such that

1. *What* denotes the target action specified in the given query such as to retrieve, insert,

delete, or update information,

2. *Where* denotes the set of data sources implied in the query,

3. *Who* denotes the set of attributes that are presented in the query,

4. *When* denotes the temporal aspect of the query (optional), and

5. *Why* denotes a description of the query (optional). !

The role of the *what*, *where* and *who* event attributes are self-explanatory in the definition. The distance between user queries with respect to time, i.e., the *when* event attribute, seems irrelevant in our context since we are focusing on detecting a single

```
MESSAGE −$ verb DIRECTOBJECT
DIRECTOBJECT −$ determiner DETERMINER1
DETERMINER1 −$ determiner DETERMINER2
DETERMINER1 −$ adverb ADVERB1
DETERMINER1 −$ adjective ADJECTIVE1
DETERMINER1 −$ noun NAME1
DETERMINER2 −$ noun NAME1
ADVERB1 −$ adjective ADJECTIVE1
ADJECTIVE1 −$ noun NAME1
ADJECTIVE1 −$ preposition INDIRECTOBJECT
NAME1 −$ . END
NAME1 −$ adjective ADJECTIVE1
NAME1 −$ adverb ADVERB1
NAME1 −$ and ENDLISTOFNAMES
NAME1 −$ adjective ADJECTIVE1
NAME1 −$ preposition INDIRECTOBJECT
NAME1 −$ comparative COMPARATIVE1
NAME1 −$ relative-pronoun CONJUNCTION1
NAME1 −$ copulative-conjunction CONJUNCTION2
NAME1 −$ relative-conjunction NAME2
NAME1 −$ determiner DETERMINER1
ENDLISTOFNAMES −$ determiner determiner1
COMPARATIVE1 −$ to NAME1
COMPARATIVE1 −$ than NAME1
CONJUNCTION2 −$ verb DIRECTOBJECT
CONJUNCTION2 −$ relative-conjunction NAME2
CONJUNCTION2 −$ relative-pronoun CONJUNCTION1
INDIRECTOBJECT −$ the THE1
INDIRECTOBJECT −$ noun NAME1
THE1 −$ table TABLE
TABLE −$ noun NAME1
NAME2 −$ noun CONJUNCTION1
CONJUNCTION1 −$ bool BOOL1
CONJUNCTION1 −$ auxiliary-verb AUXILIARYVERB
CONJUNCTION1 −$ verb VERB1
BOOL1 −$ verb VERB1
AUXILIARYVERB −$ verb VERB1
VERB1 −$ determiner DETERMINER1
VERB1 −$ adjective ADJECTIVE1
VERB1 −$ noun NAME1
```

Fig. 3. The grammar used to detect the role of each word

query event from a voice input. Note also that the *why* event attribute should be defined ideally as an unambiguous description of the user query to be useful. One way is through adopting a canonical definition of an event as many authors have considered canonical

representations of natural language [21, 22] with the goal of reducing the ambiguity in meaning. Canonical representations often require a semantic analysis which identifies the roles of entities and the relationships between the entities, and an analysis of the domain where these entities are defined. Although each user query can be described in a canonical form, the detailed description of each query and their corresponding semantic analysis are currently beyond the scope of our work. Thus, we model a user query event with three obligatory attributes: *what*, *where* and *who* leaving the other two attributes *why* and *when* optional. Finally, the meaning of a missing or unknown attribute is *any* by default in our work.

Event Detection Intuitively, a user query event is an action that responds to a user goal (**what**) performed on a set of attributes (**who**) that belong to certain database entities (where) at a particular time (**when**) with intuitive description of the query (**why**). The degree of equivalence among different query events is measured by the user's intention expressed in these event attributes. Hence, a query event recognition consists of the recognition of these event attributes in the (word:class) stream generated at the syntactic layer, i.e., (w_1, c_1), (w_2, c_2), . . ., (w_n, c_n) where w_i is a word and c_i is the syntactic class of w_i. The query event model serves as a template for a user query in this process. We present the essence of our event attribute detection approach below without presenting the complete list of linguistic rules used in our work.

– Detection of *what*: The detection of a *what* attribute is accomplished by finding the verbs that are likely used to express a query action, such as *retrieve*, *enumerate*, *show*, *calculate*, *list*, *select*, etc. We maintain a controlled set of such words in our approach.

– Detection of *who*: The *who* attribute is detected by finding the following patterns:

• *<verb>* [*<determiner>*]* *<noun>* *<preposition>*
• *<verb>* [*<determiner>*]* *<noun>* *<comparative>*
• *<verb>* [*<determiner>*]* *<ad jective>* *<noun>*

where !*verb*" is the one used to detect a *what* attribute above. If the (word:class) stream matches any of the patterns above, the *noun* word in the stream is likely the *who* event attribute.

– Detection of *where*: The word '*table*' is a strong indication for a data source in the user query. In such a case, nouns following the word '*table*' are taken as dana source names. Otherwise, those nouns which occur after the *who* nouns in a close proximity are considered as the data sources.

We demonstrate the query event detection process in the following example.

Example 4. Consider the following (word:class) stream to illustrate the detection of the *who* event attribute from it.

(Retrieve:verb),	*(the:determiner),*	*(name:noun)*	*(of:preposition)*
(the:determiner),	*(students:noun)*	*(which:preposition),*	*(take:verb),*
(the:determiner),	*(course:noun),*	*(of:preposition),*	*(cs203:noun),*
(with:preposition),	*(age:noun),*	*(greater than:comparative),*	*(twenty:noun),*
(and:copulative-conjunction),		*(that:preposition),*	*(have:verb),*
(the:determiner),		*(best:adjective),*	*(grades:noun).*

From the (word:class) stream, the following *who* attributes are detected accordingly:

1. The pattern !*noun*" !*preposition*" !*noun*" in conjunction with "course of cs203" yields the expression course = 'cs203' as the preposition 'of' is translated to '='.

2. The pattern !*noun*" !*comparative*" !*noun*" found in "age greater than 21" yields the expression: age > 'twenty'. Additional comparatives *greater than*, *less than*, *equal to*, and *different from* are interpreted as >, <, =, and ! respectively.

3. The pattern !*ad jective*" !*noun*" matches "best grades" in the word stream. In our work, this pattern yields an SQL expression invoking the user-defined function MAX(grades). However, since the meaning of an adjective frequently depends on the context, it is difficult to translate it to an SQL expression without consulting with the context. In our present work, we assume that the context is given.

User Query Specification The underlying structure of query events detected can be represented in XML. Queries expressed in XML format have the advantage of being independent of the language (whether natural or SQL language) and allow us to share query events and subsequently the embedded data in the events across different information systems. In XML, we can define the structure of query events by the following Document Type Definition (DTD):

```
<!ELEMENT QUERIES (QUERY)*>
<!ELEMENT QUERY (WHAT)>
<!ELEMENT WHAT (TEXT, WHERE+, WHEN?, WHY?)>
<!ELEMENT WHERE (TEXT, WHO+)>
<!ELEMENT WHO (TEXT)>
<!ELEMENT WHEN (TEXT)>
<!ELEMENT WHY (TEXT)>
<!ELEMENT TEXT (#PCDATA)>
```

We now show an example of how a query in natural language can be rewritten as a query event using the XML DTD presented above.
Example 5. Consider the following user query again presented in Example 1:
> *Retrieve the name, the course, and the grade of the students that have a lecturer*
> *which teach them a course.*

The event associated to this query in expressed in XML as follows.

```
<?xml version="1.0"?>
<QUERIES>
  <QUERY>
    <WHAT><TEXT>Retrieve</TEXT>
      <WHERE><TEXT>Students</TEXT>
        <WHO><TEXT>Name</TEXT>
        <WHO><TEXT>Course</TEXT>
        <WHO><TEXT>Grade</TEXT>
      </WHERE>
      <WHERE><TEXT>Lecturer</TEXT>
        <WHO><TEXT>Name</TEXT>
        <WHO><TEXT>Course</TEXT>
      </WHERE>
      <WHY><TEXT>if is(y, lecturer(y)) and is(x, student(x))
```

```
        then SELECT(<name, course, grade> in x
        such that teach(y, x))</TEXT>
      </WHY>
    </WHAT>
  </QUERY>
</QUERIES>
```

in which the *why* attribute is manually made up for a demonstration purpose.

Note that the query event DTD implies hierarchical relationships among the event attributes: the *what* attribute is the highest container which contains *where* attributes which, in turn, contain *who* attributes. The *when* and *why* attributes are optional.

As we add instances of attributes to the *what* container, the query becomes more specific. This idea is captured in Figure 4. In the figure, we make the general query goal, *retrieve*, more specific by adding two *where* attributes with values *students* and *lecturer*. The same query becomes even more specific by adding *name*, *course* and *grade* as *who* attributes. An addition of *when* and *why* will further increase the level of specificity of the query.

Moreover, if two events $e1$ and $e2$ have the same hierarchy structure, we can perform an *algebraic operation* over them to produce a new event which is likely of a different level of specificity. For example, $e1$ % $e2$ may yield the union or intersection of the two events depending on the definition of %. Figure 5 illustrates the union of the two following query operations:

– *Retrieve the id, name, and course values from the students table.*
– *Retrieve the id, course, grade and lecturer values from the students table.*

4. Prototype

We developed a software prototype, called Voice2SQL as shown in Figure 6, to demonstrate the proposed query event identification approach and tested it with a number of queries spoken in Spanish.
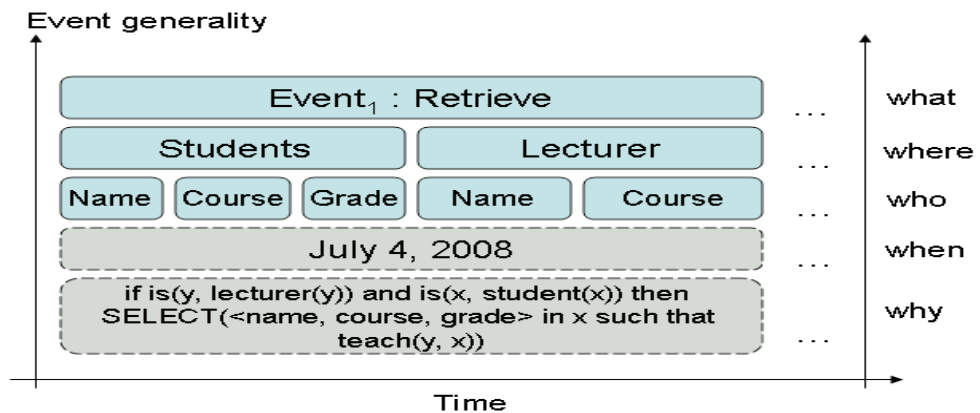

Fig. 4. The hierarchy implied among the event attributes in our event model. As more attributes are added to the general *what* attribute, the query becomes more specific.
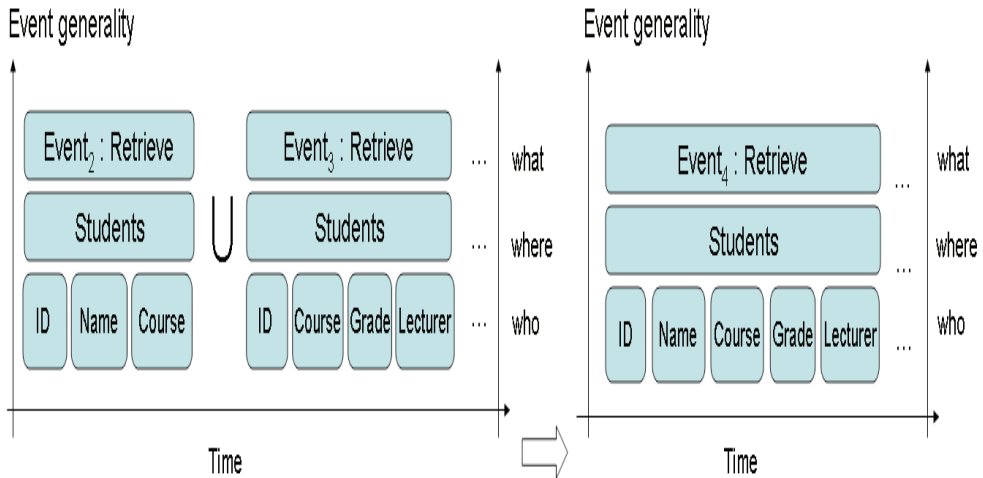
Fig. 5. Union of two query events in the proposed event model (*e*4 = *e*2 & *e*3). In this case, each event represents a query operation in the database and the result of the union of *e*2 and *e*3 is *e*4.

The upper part of Voice2SQL in Figure 6 shows the pseudo-SQL expression obtained from the natural language query in Spanish: "Calcular el nombre y la nota de los alumnos de el curso de redes cuya ciudad sea igual a Arequipa de la tabla registros" which is equivalent to "Recover the name and the grade of the students of the course of network whose city is Arequipa from the table records" in English. From the user query, Voice2SQL generates the SQL expression shown in the lower part: "CALCULAR nombre, nota WHERE curso = redes AND ciudad = arequipa AND son(alumnos) FROM registros" which is equivalent to "SELECT name, grade WHERE course = 'networking' AND city = 'arequipa' AND are(students) FROM records" in English. Additional examples of Spanish user queries that have been translated successfully by Voice2SQL are also provided in Table 2 in English.

5. Conclusion

We have presented in this chapter an approach to translate queries in spoken natural language to queries in SQL to increase the level of human-computer interaction in database access. Our focus in developing such an approach was to extend the concept of user queries as the presence of query events in the user speech. A formal query event model is presented together with a software prototype of the proposed approach. The proposed event model finds a correspondence between queries in a spoken natural language and the SQL language by studying the common linguistic components that are present in both languages. The proposed approach was moderately evaluated by developing a software prototype. A rigid, large scale evaluation is necessary to validate the benefit of the proposed event model and query event identification.
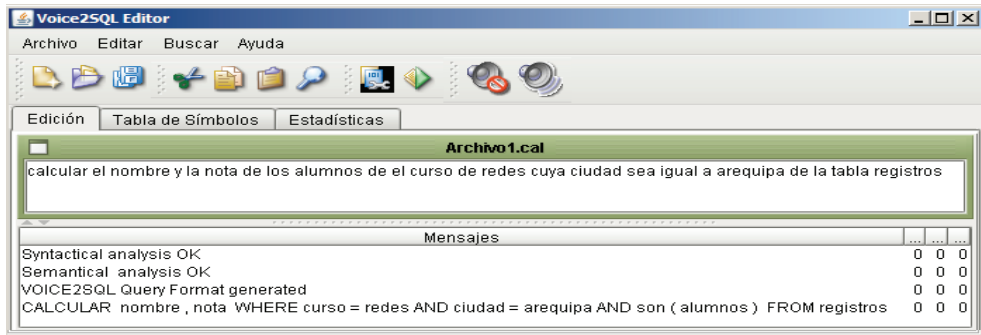
Fig. 6. Voice2SQL GUI. The upper part shows the input user query and the lower part shows the pseudo-SQL expression produced by Voice2SQL.

Spoken user query	SQL generated by Voice2SQL
Calculate the average mark in the students of the course of AI	SELECT AVG(Mark) FROM students WHERE course = 'AI';
Select the students of fifth cycle that are not registered in the course of AI	SELECT * FROM students WHERE course <> 'AI' AND cycle = 5;
Enumerate the approved students in the course of AI	SELECT * FROM students WHERE course = 'AI' AND approved(students);
Select the students of fifth cycle that are not registered in the course of AI	SELECT * FROM students WHERE course <> 'AI' AND cycle = 5;

Table 2: Query examples processed by Voice2SQL.

6. References

T. Kahveci and A. K. Singh, "Efficient index structures for string databases," in *Proceedings of the 27th International Conference on Very Large Data Bases*, San Francisco, CA, USA, 2001, pp. 351–360.

Y. Li, H. Yang, and H. V. Jagadish, "NaLIX: A generic natural language search environment for XML data," *ACM Transactions on Database Systems*, vol. 32, no. 4, p. 30, 2007.

A. Witkowski, S. Bellamkonda, T. Bozkaya, N. Folkert, A. Gupta, J. Haydu, L. Sheng, and S. Subramanian, "Advanced SQL modeling in RDBMS," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 83–121, 2005.

P. Merlo and S. Stevenson, "Automatic verb classification based on statistical distributions of argument structure," *Computational Linguistics*, vol. 27, no. 3, pp. 373–408, 2001.

S. Polyviou, G. Samaras, and P. Evripidou, "A relationally complete visual query language for heterogeneous data sources and pervasive querying," in *Proceedings of the 21st International Conference on Data Engineering*, April 2005, pp. 471–482.

R. Harrathi and S. Calabretto, "A query graph for visual querying structured documents," in *Proceedings of the 2nd International Conference on Digital Information Management.*, vol. 1, 2007, pp. 116–120.

L. Aversano, G. Canfora, A. D. Lucia, and S. Stefanucci, "Understanding SQL through iconic interfaces," in *Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life*, Washington, DC, USA, 2002, pp. 703–710.

T. Catarci and G. Santucci, "Diagrammatic vs textual query languages: a comparative experiment," in *Proceedings of the 3rd IFIP Working Conference on Visual Database Systems 3 (VDB-3)*, London, UK, UK, 1995, pp. 69–83.

S. Harada, J. O.Wobbrock, and J. A. Landay, "Voicedraw: a hands-free voice-driven drawing application for people with motor impairments," in *Proceedings of the 9th international ACM SIGACCESS Conference on Computers and Accessibility*, New York, NY, USA, 2007, pp. 27–34.

C. D. Wickens and J. G. Hollands, *Engineering Psychology and Human Performance (3rd Edition)*. Prentice Hall, September 1999.

"Merriam-Webster dictionary," http://www.merriam-webster.com, 2008.

R. C. F. Tucker, M. Hickey, and N. Haddock, "Speech-as-data technologies for personal information devices," *Personal and Ubiquitous Computing*, vol. 7, no. 1, pp. 22–29, 2003.

H. Xie, P. Andreae, M. Zhang, and P. Warren, "Learning models for English speech recognition," in *Proceedings of the 27th Australasian Conference on Computer Science*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2004, pp. 323–329.

D. Ponceleon and S. Srinivasan, "Structure and content-based segmentation of speech transcripts," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2001, pp. 404–405.

G. Antoniol, V. F. Rollo, and G. Venturi, "Linear predictive coding and Cepstrum coefficients for mining time variant information from software repositories," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1–5, 2005.

E. H. C. Choi, "On compensating the Mel-frequency Cepstral coefficients for noisy speech recognition," in *Proceedings of the 29th Australasian Computer Science Conference*, Darlinghurst, Australia, Australia, 2006, pp. 49–54.

H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE Transactions on Speech and Acoustics*, vol. 2, pp. 587–589, October 1994.

T. Ganchev, N. Fakotakis, and G. Kokkinakis, "Comparative evaluation of various MFCC implementations on the speaker verification task," in *Proceedings of the International Conference Speech and Computer*, vol. 1, 2005, pp. 191–194.

O. U. Florez and S. Lim, "A spoken natural language-based interface for querying SQL databases," *International Journal of Information Technology and Intelligent Computing (to appear)*, 2008.

J. Zhu, "Towards scalable flow and context sensitive pointer analysis," in *Proceedings of the 42nd Annual Conference on Design Automation*. New York, NY, USA: ACM, 2005, pp. 831–836.

S. Wintner, "Formal language theory for natural language processing," in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2002, pp. 71–76.

S. Nirenburg and V. Raskin, "Ontological semantics, formal ontology, and ambiguity," in *Proceedings of the International Conference on Formal Ontology in Information Systems*. New York, NY, USA: ACM, 2001, pp. 151–161.

Human Computer Interaction: New Developments

Edited by Kikuo Asai

The book consists of 20 chapters, each addressing a certain aspect of human-computer interaction. Each chapter gives the reader background information on a subject and proposes an original solution. This should serve as a valuable tool for professionals in this interdisciplinary field. Hopefully, readers will contribute their own discoveries and improvements, innovative ideas and concepts, as well as novel applications and business models related to the field of human-computer interaction. It is our wish that the reader consider not only what our authors have written and the experimentation they have described, but also the examples they have set.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Omar U. Florez and SeungJin Lim (2008). Modeling Query Events in Spoken Natural Language for Human-Database Interaction, Human Computer Interaction: New Developments, Kikuo Asai (Ed.), ISBN: 978-953-7619-14-5, InTech, Available from:
http://www.intechopen.com/books/human_computer_interaction_new_developments/modeling_query_events_in_spoken_natural_language_for_human-database_interaction

INTECH
open science | open minds