An Intelligent Natural Language Interface to Relational Databases

Anh Kim Nguyen, Phuong Hong Nguyen

Abstract—This paper is a study on constructing a natural language interface to relational databases, which accepts natural language fuzzy questions as inputs and generates answers under the form of tables or short answers. Firstly, the question is parsed using a semantic grammar and then, it is translated into a SQL query using a set of translation rules. Finally, a database management system is left to find the result table with its own specialized optimization and planning techniques. Experimental results show that this approach can analyze a wide range of questions with high accuracy and produce reasonable answers.

Index Terms—database, fuzzy question, natural language interface, semantic grammar.

I. INTRODUCTION

Database management systems (DBMSs) have been widely used because of their efficiency in storing and retrieving data. However, traditional query language, as a standard database user interface, has often frustrated database user including expert users by enforcing rigidity and preciseness in query writing and returning query results exactly what are being asked. Such a direct query answering method requires the user to have a detailed knowledge of both the database schema and the query language and the user also has to know exactly what information to search. This method is inconvenient for non-experienced and casual users because they often cannot construct a query intelligently and properly. So, it is very useful to provide users with intelligent interfaces to databases, which have ability to understand natural language queries and more important, have ability to interpret and evaluate imprecise criteria in queries. Including vague criteria in a database query may have two advantages:

- the flexibility of the query expression
- the possibility to refine the results, assigning each tuple to the corresponding degree of criteria satisfaction.

We particularly focus on the possible vagueness of the selection criterion and the fuzzy quantity, which involve certain vague terms, currently used in natural spoken language.

In a vague query, the selection criterion and the quantity are no longer Boolean, so it can be more or less satisfied by the database tuples. Therefore, for each table row, a satisfaction degree is estimated, which stands for a measure of its compatibility with the vague criterion.

The fuzzy - set theory is already established as the adequate framework to model and manage vague expressions, or in other words, to evaluate vague queries which are sent to relational databases ([3]).

In the context of fuzzy query language, many authors proposed an extension to relational algebra in order to develop a fuzzy SQL that provides the means for performing queries with some uncertain concepts and obtaining answers ([3], [4]). We developed an interface that allows us to make questions in flexible natural language without having to modify either the structure of the database or the DBMS query language. That is, we developed an intelligent interface and not a fuzzy query language.

The researches on natural language interface to databases (NLI2DBs) have recently received attentions from the research communities ([1], [5], [10]). The purpose of natural language interfaces is to allow users to compose questions in natural language and to receive responses under the form of tables or short answers. Since natural language always contains ambiguities, most NLI2DBs are implemented in a specific domain and can only understand a subset of a natural language.

In this paper, a semantic grammar (SG) is developed to parse the syntactic and semantic structures of user queries. This grammar can deal with natural language fuzzy queries for all retrieved purposes. A more difficult task is to translate the syntactic-semantic trees of natural language fuzzy queries into SQL queries and to evaluate these fuzzy queries.

This paper presents our approach to the NLI2DB. Our implemented system includes two main modules: (i) a Query Interpreter (QINTER) interpret a natural language question to a standard tree; and (ii) a Query Translator (QTRAN) translate a standard tree to a SQL query. QINTER uses a domain-independent semantic grammar and a CYK parsing algorithm to parse user questions and then QTRAN uses a set of translation rules to translate the standard trees of user questions into SQL queries. To test the feasibility of the system, a specific database system - a student management database system in Vietnamese – is used. The proposed system architecture guarantees its portability across domains.

The remaining sections of this paper are organized as follows. Section II introduces our technique to interpret natural language questions to a standard tree. Section III describes our method to translate the standard trees of user questions into SQL queries. Our experimental results are discussed in Section IV. In the end, Section V concludes the paper and proposes possible future works based on this approach.
II. QUERY INTERPRETER

Most existing NLI2DBs are quite rigid in interpreting natural language queries. They just look for keywords in the sentence [5] or using some templates in analyzing the user’s input [10]. Such approaches cannot deal with questions in unpredictable formats. Most NLI2DBs are domain-dependent, as they require predefined knowledge of the working domain in constructing templates or semantic rules ([5], [10]). In this paper, we present our query interpreter named QINTER - a module that uses a restricted domain-independent semantic grammar to interpret Vietnamese questions to syntactic-semantic trees. In this system, users can type natural language fuzzy queries without following any predefined template. Thus, the fuzzy information in queries must be stored in the data dictionary and could be used when it is necessary. The fuzzy information and the semantic grammar are introduced below.

A. The Fuzzy Information in Queries

The vague criterion in a database context is an expression standing for rows selection condition in a database query which include vague terms. The vague terms usually refer linguistic values from the attribute linguistic domains. Other vague terms coming from natural language may be included in the database query and must be taken into account in query evaluation. Some of them are linguistic modifiers and linguistic comparators. Staying close to a linguistic value, the linguistic modifiers affect the intensity of this one. There are some categories for the linguistic modifiers with different semantic effects: concentration (more, very, extremely) and dilatation (less, greatly, remotely). The semantic effects of these modifiers are domain-independent. A linguistic comparator applicable to crisp number values or fuzzy numbers is approximately equal.

Fuzzy quantity usually refers fuzzy quantifiers [4]. Fuzzy or linguistic quantifiers allow us to express fuzzy quantities or proportions in order to provide an approximate idea of the number of elements of a subset fulfilling a certain condition (absolute fuzzy quantifiers) or the proportions of this number in relation to the total number of possible elements (relative fuzzy quantifiers). Some absolute fuzzy quantifiers are ‘close to 7’, ‘around 7’ and relative fuzzy quantifiers are ‘the majority’ or ‘most’ and so on.

These linguistic values, linguistic modifiers, linguistic comparator and fuzzy quantifiers are fuzzy information in queries. The linguistic values, linguistic comparator and absolute fuzzy quantifiers are domain-dependent vague terms. The modifiers and relative fuzzy quantifiers are domain-independent vague terms.

All this fuzzy information must be stored in the data dictionary and they will be used when it is necessary.

B. The Semantic Grammar

In general, each natural language question uses a particular grammatical structure. Each position in the question is used for a specific purpose, such as storing an object asked by the user, an object’s attribute, a value, or a linkage between objects. The questions can be in the following formats:

(i) Questions start with an instructive verb (e.g., “Đưa ra show/tìm find/liệt kêlist/……”)
(ii) Questions start with an interrogative pronoun (e.g., “Ai who/Cái giWhat/……?”)
(iii) Questions with the structure <Danh ngữ phrase> … là is <đặt tu nombre ván interrogative pronoun>
(iv) Questions with the structure <Danh ngữ phrase> … là/để cóhas <cum từ phrase> phải khôngisn’t it?
(v) Questions with the structure <cum từ phrase> … phải khôngisn’t it?
(vi) Questions with the structure Có di phải is true that <cum từ phrase> … phải khôngisn’t it?
(vii) Questions whose interrogative pronoun (e.g., “Ai who/Cái giWhat/……?” ) can appear in any position of the question.

By analyzing a large set of user questions, we found that although user questions are expressed in various ways, they always follow a specific organizing principle. Therefore, it is possible to define a semantic grammar that covers all types of unrestricted NL queries. Authors of [9] constructed a set of restricted, domain dependent query formats. Especially, our rule set is also capable of handling fuzzy quantified questions with fuzzy terms. These questions are hardly representable by a query language supported by a DBMS. This grammar is strong enough to cover a large number of questions that are often given by users. It consists of 86 rules, using 49 non terminal symbols and 47 terminal ones. Some examples of our semantic rules are shown below:

1. <conditions> → <selection condition><conjunction><conditions>
2. <conditions> → <joint condition><conditions>
3. <selection condition> → <crisp selection condition>/<fuzzy selection condition>
4. <fuzzy selection condition> → <fuzzy value>
5. <joint condition> → <source> <negative> <SR>
6. <source> → <quantity><entity><conditions>
7. <quantity> → <stress word><number>/ <fuzzy quantifier>

To interpret a user’s question to a standard tree, two processes are carried out: (i) parsing the user’s question using the semantic grammar; and (ii) interpreting the syntactic-semantic tree of the user’s question. These processes are described in sections C and D below.

C. Parsing a User’s Question

First, the parser detects words and their semantic categories in the user’s question using a word-category dictionary. Then, the parser derives syntactic-semantic trees from the input question using the semantic grammar introduced in Section B. Since the proposed grammar is context free and is not in the Chomsky normal form, an improvement of the Cocke-Younger-Kasami (CYK) algorithm is used to parse the input question.

As our grammar is context free and domain independent, the number of generated trees may be large. Meaningless trees or...
trees those do not represent the user’s intention should be removed from the output using all possible constraints from the input question. These constraints are the correspondence between attributes and entities, between attributes and values (e.g., a date should accompanies with the ‘date’ attribute), etc. Then, if the input question contains semantic ambiguities, this dictionary by the following information: (i) attribute’s name in Vietnamese; (ii) attribute’s name in the database; (iii) name of the entity/table containing this attribute. Information about a semantic role includes (i) role’s name in Vietnamese; (ii) entity1’s name; (iii) entity2’s name; and (iv) table’s name in the database corresponding to this semantic role.

QINTER generates a multi-choice question asking the user to pick the closest one to his/her intention. The syntactic-semantic trees that fit with the user’s intention are selected as the output for the parser.

D. Interpreting the Syntactic-Semantic Tree of the User’s Question

The syntactic-semantic trees generated by the previous process satisfy all possible constraints in the database. However, the semantic relations are not explicitly and completely presented in the trees. This process deals with this problem. It analyzes semantic relations in these trees and transforms them into another format that is closer to the SQL syntax. The trees after being transformed are called standard trees [6]. This process uses a data mapping dictionary that provides information about the database structure including entities, attributes, and semantic roles (SR). For example, an attribute is described in

An example of the output tree generated by the QINTER is shown in Fig 1.

Finally, QINTER rephrases the input question based on the standard tree and displays it to the user. If the rephrased question expresses exactly the user’s intention, (s)he will click a button forcing QTRAN to generate the SQL-query from the standard tree.

For example, the rephrased question of the question in Fig. 1 is “Tim các sinh viên có điểm học hậu hết các môn do giáo viên A dạy” (“Find the students who have good mark and study most subjects taught by lecturer A”).

Fig.1. The standard tree for the question “Tim các sinh viên giỏi học hậu hết các môn do giáo viên A dạy” (“Find the good students who study most subjects taught by lecturer A”).

1 For a detailed description of this technique, see [6].
III. QUERY TRANSLATOR

A. Fuzzy Knowledge Representation in Relational Databases

As we have seen in the previous section, the fuzzy terms are stored in the data dictionary. To translate the standard tree of user question to an SQL query, the knowledge of these terms must be contained in the system’s knowledge base and must be stored in an accessible manner by the System.

According to fuzzy set theory, the linguistic values in fuzzy selection conditions are defined as trapezoidal fuzzy sets [12]. The membership functions of the trapezoidal fuzzy sets are characterised by four parameters (a, b, c, d), consists of three consecutive segments. The application of dilatation/ concentration modifiers should increase/decrease the size of these segments and, therefore, be implemented with the modification kept in proportion to the center of the full membership segment. For a given fuzzy set, the modified fuzzy set with a concentration modifier (a dilatation modifier) is defined by parameter β which controls the shrinking degree of a concentration modifier (the relaxing degree of a dilatation modifier). So, β of a concentration modifier (a dilatation modifier) must satisfy

\[
0 < \beta < 1 \text{ (} \beta > 1\text{)}.
\]

The linguistic comparator defined as triangular fuzzy sets [12]. The fuzzy quantifiers can be absolute or relative in which absolute fuzzy quantifiers are defined as triangular fuzzy sets and relative as trapezoidal fuzzy sets.

The membership functions of the triangular fuzzy sets for the linguistic comparator and absolute fuzzy quantifiers are characterised by three parameters (a, b, d) where a = b−α, d = b+α with b is the number which accompany with these fuzzy terms and α is the parameter of linguistic comparator and absolute fuzzy quantifiers which does’nt depend on number b. We will organize the knowledge using tables or relations.

The tables stored in the the system’s fuzzy knowledge base are the following: table FuzzyAtt contains the information about linguistic values of attributes together with four parameters (a, b, c, d); table Modifiers contains the information about modifiers together with its parameter; table FuzzyComp contains the information about linguistic comparators of one attribute together with its parameter b; table RFuzzyQuan contains the information about relative fuzzy quantifiers together with four parameters (a, b, c, d); table AFuzzyQuan contains the information about absolute fuzzy quantifiers of one attribute in one entity together with its parameter α.

B. Evaluating Fuzzy Criterion

Some different approaches have been proposed for expressing and solving flexible queries [3][4]. At present time, SQLf is the most complete SQL extension using Fuzzy set [11]. SQLf is the only known proposal of flexible querying system allowing linguistic quantification over set of rows in queries throw the extension of SQL nesting and partitioning structures with fuzzy quantifiers [11].

For query evaluation, SQLf used the derivation evaluation mechanism with a threshold t. The main idea of the evaluation mechanisms is to allow adding fuzzy querying capabilities on top of an existing RDBMS. At the moment, it is not clear that the derivation is applicable to all constructions of SQLf, particularly, it is doubted of its applicability in queries using fuzzy quantifiers.

In this paper, applying the derivation evaluation mechanism, we construct a set of translation rules for all possible structures in the standard trees of user questions to translate the standard trees to an SQL query.

In a vague context, the compound criterion is a logical expression which can contain fuzzy comparisons and fuzzy quantities. The logical operators of this logical expression are extended to fuzzy aggregation connectives. We simply use the minimum and maximum functions for the conjunctive and disjunctive connectives, and the complement for the negation connective.

C. Translating to SQL

The translating process traverses an input tree in the bottom up direction. In this process, each Source node is considered as a subtree, in which Entity nodes and Conditions nodes are translated as sub-SQL queries. After the root node ‘Query’ has been traversed, a complete SQL is generated. It is clear that if translation rules for all structures of Source and Query nodes is provided, it is possible to translate any standard tree to an SQL query. Such a set of 52 translation rules is proposed and used in QTRAN. Some of our translation rules are shown below:

1. SL(E <crisp selection condition>) = select KE from E where <crisp selection condition>

2. SL(E < Attr Label >) = select KE, m(Attr, a,b,c,d) as Degree from E, FuzzyAtt where FAttr = ‘Attr’ and LLabel= ‘Label’ and Attr <> a and Attr <> d

3. SL(S SR’ CrispSQL) = select distinct KS from SR’ where KR in CrispSQL

4. SL(S ‘phù dân’ <SR> R) = select KS from S where KS not in (select KS from SR where KR in (select KR from R))

5. SL(FuzzySQL <SR> <RQ> CrispSQL) = select KS, min(S.Degree, T.Degree) as Degree from FuzzySQL as S, (select KS, h(Num1, Num2, a,b,c,d) as Degree from (select KS, count(KR) as Num1 from S, CrispSQL as R where S.KR = R.KR group by KS ) as V, RFuzzyQuan, (select count(KR) as Num2 from CrispSQL as R) as G where RQuan = ‘RQ’) as T where S.KS = T.KS

In the above rules:

- SL(X) represents for a SeLect translation rule;
- SR is the table’s name in the database corresponding to this semantic role <SR>.
- FuzzySQL is the result of applying translation rules for fuzzy structures in the standard trees of user questions, otherwise CrispSQL.
- Degree is a special column which is inserted into result table of FuzzySQL and is used to measure the satisfaction degree for each row of this table.
• \(K_S, K_R, K_E\) are keys of entities \(S, R,\) and \(E\), respectively. (\(E\) can be the query’s target).
• \(m(Attr, a,b,c,d)\) is a user-defined function which is used to estimate the satisfaction degree of value of attribute \(Attr\) with linguistic value \(Label\) characterised by four parameters \((a, b, c, d)\).
• \(h(Num1, Num2, a,b,c,d)\) is a user-defined function which is used to estimate the satisfaction degree of value \(Num1/Num2\) with relative fuzzy quantifier \(RQ\) characterised by four parameters \((a, b, c, d)\).

**Clause 1**: The above rules do not modify the semantic meaning of structures in trees whose root node is ‘Source’ or ‘Query’.

It is easy to find out that for the standard tree that satisfies the above-stated conditions, the semantic of the structure of ‘Source’ and ‘Query’ node on the left can be verified to be the same as the corresponding structure on the right using the translation rule \(S\). The following CrispSQL is generated by applying Rule 1 and 3 for the subtree \(T\) in Fig. 1:

\[
SL(subtree\ T) = \\
SELECT\ subjectID\\
FROM\ Teach\\
WHERE\ lecturerID\ IN\\
SELECT\ lecturerID\ FROM\ Lecturer\\
WHERE\ lectname=’A’
\]

The following FuzzySQL is generated by applying Rule 2 for the subtree \(S\) in Fig. 1:

\[
SL(Student < Mark Good >) = select\ studentID , m(Mark, a,b,c,d)\ as\ Degree\ from\ Student, \ FuzzyAtt\ where\ FAttr = \ ‘Mark’\ and\ LLabel= \ ‘Good’\ and\ Attr < a\ and\ Attr <> d
\]

Finally, applying Rule 5 for the the standard tree in Fig. 1 with above FuzzySQL and CrispSQL, the following complete FuzzySQL is generated:

\[
SL(FuzzySQL < Study > <most> CrispSQL) = select\ studentID,\ min(S.Degree, T.\ Degree)\ as\ Degree\ from\ FuzzySQL\ as\ S,\ (select\ studentID,\ h(V.Num1, G.Num2, a,b,c,d)\ as\ Degree\ from\ (select\ studentID,\ count(subjectID)\ as\ Num1\ from\ Study,\ CrispSQL\ as\ R\ where\ Study.subjectID = R.studentID\ group\ by\ studentID )\ as\ V, \ RFuzzyQuan,\ (select\ count(subjectID)\ as\ Num2\ from\ CrispSQL\ as\ R)\ as\ G\ where\ RQuan = \ ‘most’)\ as\ T\ where\ S.\ studentID = T.\ studentID
\]

**IV. EXPERIMENTAL RESULTS**

To evaluate the system performance, we carried out experiments with a student management database. The evaluation focuses on two aspects: (i) the accuracy of rephrased questions from user questions; and (ii) the quality of top-k answers. These aspects were evaluated independently by us.

20 users were offered to participate in our experiments. Their tasks were to formulate questions on this database without knowing the database structure and to evaluate system responses. Since these participants do not know the SQL syntax, they cannot evaluate directly SQL queries. Instead, they were asked to evaluate the rephrased questions derived in the second process of QINTER (see Section II.D). As mentioned in Clause 1 (Section C of III), the SQL query does not modify the semantic meaning of structures in trees whose root node is ‘Source’ or ‘Query’. Therefore, we can guarantee that if the rephrased question is correct, the SQL query generated by QTRAN is also correct.

The experiments were conducted by testing questions with various lengths and with different characteristics: crisp or fuzzy, positive or negative, quantified or unquantified. Each question was stated in several ways in order to test the robustness of the system. We got user feedbacks by asking them to answer the following questions:

- Do you think that the rephrased question is fluent (Y/N)?
- Do you think that the system rephrased correctly your intention (Y/N)?

The experimental results with QTRAN are shown in Table 2. 309 rephrased questions in total were used in the experiments. 91.91% rephrased questions are correct, among which 80.26% rephrased questions are fluent. It indicates that QTRAN is quite accurate and robust in translating user questions.

**Table 2. Experimental Results with QTRAN**

<table>
<thead>
<tr>
<th>Question’s characteristics</th>
<th>Rephrased questions</th>
<th>Total questions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correct</td>
<td>Incorrect</td>
</tr>
<tr>
<td></td>
<td>Fluent</td>
<td>Unfluent</td>
</tr>
<tr>
<td>Crisp Positive</td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>Unquantified</td>
<td>66</td>
<td>1</td>
</tr>
<tr>
<td>Quantified</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>Negative</td>
<td>42</td>
<td>7</td>
</tr>
<tr>
<td>Quantified</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>Unquantified</td>
<td>43</td>
<td>8</td>
</tr>
<tr>
<td>Quantified</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Negative</td>
<td>29</td>
<td>8</td>
</tr>
<tr>
<td>Total questions</td>
<td>248</td>
<td>36</td>
</tr>
<tr>
<td>%</td>
<td>80.26</td>
<td>11.65</td>
</tr>
<tr>
<td>%</td>
<td>91.91</td>
<td></td>
</tr>
</tbody>
</table>

**V. CONCLUSIONS AND FUTURE WORKS**

In this paper, we introduced our approach to intelligent natural language interfaces. The current prototype of our system has achieved the following results:

- A friendly interface has been provided to users: users can type natural language fuzzy questions that do not need to follow any predefined form and receive feedbacks under the form of short answers. It is suitable for people that have little or no knowledge of the database structure being used.
• The system accepts quantified questions and negative questions, which are very difficult to express in SQL syntax by naïve users.
• The system can assist users to rephrase questions correctly to his/her intention.
• Some questions (such as incomplete queries) can be automatically corrected without asking users to pick a choice.

The system is portable to other domains. When applying to other domains, we only need to modify the domain-dependent dictionaries and knowledge sources.

To improve the system performance and to provide intelligent answers, future works include:
• Enriching the knowledge sources of the system to increase the system efficiency
• Implementing techniques to relax failing fuzzy queries which their fuzzy selection condition contain linguistic values using linguistic modifier-based approach
• Implementing techniques to relax failing crisp quantified queries using linguistic quantifiers
• Researching methods to improve the coherence and the fluency of output texts

REFERENCES