An Intelligent Conversational Agent Approach to Extracting Queries from Natural Language

Karen Pudner[1], Keeley Crockett[1] *Member IEEE*, Zuhair Bandar[1]

***Abstract*—This paper is concerned with the application of a conversational agent and expert system to provide a natural language interface to a database. Typically, natural language database interfaces (NLDI's) use grammatical and/or statistical parsing. Conversational agents take a different approach, capturing key elements of user input which then trigger pre-determined output templates. It is assumed that the type of natural language questions which could be asked of a specific relational database will contain a limited number of key words (attributes), which could be captured by a conversational agent. In the proposed system, once a conversational agent has identified all relevant attributes and their values, an expert system would then apply rule based reasoning on these attributes to construct an SQL query. The knowledge base of the expert system would contain information on the database structure (metadata) and on the different possible structures of SQL queries. This would result in a real time system, which could extract both database attributes and attribute values from the user input and automatically apply a rule based reasoning system to determine the answer the user's query.**

***Index Terms*—Conversational Agents, Natural Language, SQL**

I. Introduction

Typically, natural language database interfaces use parsing to identify and categorise the user's input [1]. The parsing process requires the creation of a lexicon, which includes both database specific and non-database specific terms. Different natural language database interfaces (NLDI's) use different methods to create the database specific terms. Microsoft English Query includes an authoring tool, so that database users can populate the lexicon with synonyms and alternative phrasings for database elements such as relation and attribute names [2]. Additionally, words which define the relationship between different database elements can be added (i.e. Customers **buy** products). Masque/SQL takes a similar approach, helping users to define the relationships between database elements using an "is-a" hierarchy [3]. English Language Frontend (ELF) [4] and Precise [5] find database specific terms by extracting information on the schema and entity names from a database, and using a dictionary to identify possible synonyms.

In all these cases, once user input has been parsed, the relevant elements of user input are identified and related to the corresponding database elements. Various algorithms, with rules relating to database structure and/or SQL syntax, are then used to transform these elements into a valid SQL query. None of these NLDI's are widely used due to the fact that generally grammatical parsers do not deal well with incorrect grammar and sentence fragments, both of which may be present in user input. Also, these NLDI's have, at best, only a limited capability for allowing interaction with the user, when the user input is unclear or ambiguous.

Conversational agents take a different approach to the processing of natural language [6..11]. Rather than parsing input through the use of a lexicon, a set of scripts is used to capture specific attributes and their values from the user input. Furthermore, it is possible for conversational agents to be used in more purposeful ways, so that the capture of different attributes will initiate different actions, such as writing specific data to a file, starting up another program etc. This paper proposes a novel framework which utilizes a conversational agent as a NLDI. The conversational agent is used as a means to capture attributes from user input, (i.e. database attributes and their associated values, aggregate terms etc). The identification of these attributes will trigger the start up of an expert system, which will then transform these attributes into a valid SQL query. The expert system will contain knowledge of the database structure as its domain facts, and knowledge of SQL syntax in its rule base. A conversational agent will have the advantage over traditional NLDI's of being able to deal with ungrammatical input. It will also be able to use its conversational capabilities to guide a user through a process rather than just extracting attributes from the conversational dialogue.

The paper is organised as follows: Section II will outline the system's architecture; Section III will discuss the syntax of SQL, and outline the type of SQL query covered by the paper's remit; Section IV will discuss the methods used to collect typical questions asked by database users and outline the testing carried on the system; Section V will analyse the results and section VI will discuss the findings. Finally section VII will draw conclusions and highlight areas of future work.

II. System Architecture

A. System Overview

The proposed system, shown in Fig. 1., will consist of a conversational agent (including user interface), an expert system, a database, and interfaces between these three components and a control module.

[1]The Intelligent Systems Group, Department of Computing and Mathematics, The Manchester Metropolitan University, Chester Street, Manchester, M1 5GD, UK (phone: +44 161 247 1497; fax +44 161 247 1483; corresponding author email K.Crockett@mmu.ac.uk).

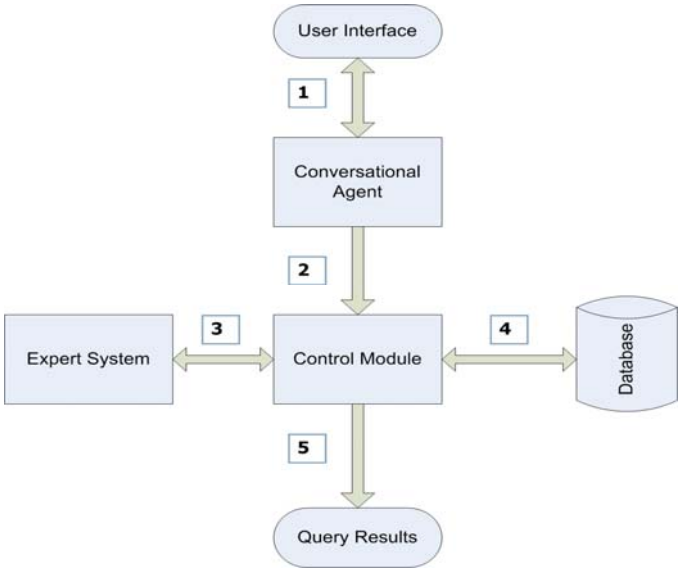

Fig. 1. System components and the order in which processes take place

1. User enters a natural language statement into the user interface, which is passed to the conversational agent.
2. Attributes and their associated values are extracted from the user input by the conversational agent, and passed to the control module.
   - If no attributes can be identified the conversational agent will engage in a dialogue to guide the users through a process aimed at capturing an attribute and its value.
   - If the user is uncooperative and fails to cooperate with the conversational agent after several attempts then the session will end.
3. The control module passes the attributes and their values to the expert system. Rule based reasoning is used to construct an SQL statement, which is then passed back to the control module.
4. The control module executes the SQL statement on the database, which returns the results.
5. The control module opens a new window and displays the query results.

Each component will now be described.

B. Conversational Agents

The idea that a computer could actually engage in a conversation with a human being was thought to be the subject of science fiction for many years. That was until British mathematician and code-breaker Alan Turing published a seminal paper, Computing Machinery and Intelligence which discussed the question "Can machines logically process information?"[12]. Since then the ability to create a computer that could understand and communicate using natural language has been the main thrust of scientists worldwide. This has led to the development of conversational agents, computer based agents that can participate in natural human dialogue with a user. The implication of this technology, even whilst still in its infancy is that a machine rather than a human operator can engage in a conversation with a person to try and solve their problem(s). The best known early conversational agent was Eliza [10]. Eliza's main trick was to use questions to draw a conversation out of the user with the agent having to perform little contribution. However the main criticism of ELIZA as a model for artificial intelligence was that it focused on the program's lack of an internal world model that could influence and track conversation [6]. Since then many conversational agents have been developed such as PARRY [7], ALICE[8]and Convagent [9].

In the context of the proposed architecture, the role of the conversational agent would be to look for each possible relevant attribute and any associated value in the user input (i.e. words relating to database attribute names, values, SQL terms etc). It will do this by creating a specific script for the capture for each attribute or combination of attributes which could be present in the user input. When user input is passed from the user interface to the conversational agent, all the different scripts will be called in turn. Each script will determine whether a specific attribute and it's associated value is present in the input, and if it is, capture it and assign it to a set of variables. These variables will then be passed on to the expert system. The system will use a specific existing conversational agent [9,11]. However, any conversational agent could be used in the proposed architecture, providing it has the ability to capture attributes.

C. Expert System

Once the conversational agent has captured the attributes present in user input, it will pass them on to an expert system. The role of the expert system will be to transform these attributes into an SQL query, using rule based reasoning. The expert system will contain

- database metadata describing the current domain
- a rule base
- an inference engine,

which will work together to create an SQL statement from the attributes. The rule base and inference engine are based on elements of an expert system described in Bigus & Bigus [13].

In order for an expert system to construct an SQL query, the rule base will need to contain some knowledge of SQL syntax. SQL was chosen as it is the language most widely used for the creation and manipulation of relational databases [14]. Section III will provide more detail about the SQL syntax and types of SQL query which the system was set up to deal with.

Knowledge of the database schema will be built into the expert system by making elements of the database, such as relations and database attributes into objects. For example, a database attribute will have data variables such as name, data type, foreign/primary key, which relations(s) it belongs to etc. Relations will have details of other relations which they are linked to, and which database attributes they contain. Once all database attributes, values, comparative terms etc. have been identified, rule based reasoning will be used to determine how these attributes could make up an SQL query. For example, knowing database attributes and values will determine which relations need to be included. Knowing

about how relations are linked will allow join clauses to be written. "WHERE" clauses will be made up of a database attribute, a comparison term and a value from that database attribute, or a numeric or date value (which means that the database attribute will need to be of the same data type). The rule based reasoning strategy is based on the concepts of forward chaining, so that once certain facts have been established about user input, these facts can then trigger the firing of other rules. The key elements of the rule construct used in the expert system are shown in (1), which is an example of one of the rules present in the rule base. This particular rule ensures that, if both an aggregated and non-aggregated attribute(s) are present in the SELECT clause of the SQL query, the non-aggregated attribute(s) is also included in a GROUP BY clause:

IF the user input contains an aggregated attribute and
a non-aggregated attribute(s),
THEN add all non-aggregated attributes to the Group
By clause

(1)

D. Database

For the purpose of this work, a relational database was implemented using MySQL. However, the architecture has been designed to incorporate the use of any database management system. The Northwind Traders sample database which is supplied with Microsoft Access 2003 was taken as a model, as the sales domain is one which is commonly used in real life. However, in order to keep the system relatively simple, the number of relations was reduced and the number of database attributes in each relation was reduced. In this way, the database could simulate a real life situation, but focus on just the key elements. The database which was implemented comprised of the following relational schema:

- customer (customer_no, customer_name, company_type, country)
- product (product_no, product_name, product_type, product_price, number_in_stock)
- order (order_no*, product_no*, no_of_products, total_price)
- total_order (order_no, customer_no*, date_ordered)

E. Control Module

The control module will detect when the conversational agent has processed the user input and will then pass the information, (e.g. database attribute names, database values, aggregate terms etc.) on to the expert system. Once the expert system has produced an SQL query, this will be passed back to the control module, which will execute the query on the database. It will also handle the display of database results in a new window

III. Analysis of SQL Queries

A. Overview of SQL

SQL statements can be divided into 5 main types; data retrieval and data manipulation (DML), data definition (DDL), transaction control and data control language (DCL). Data retrieval and data manipulation cover the types of procedure which are commonly performed on databases by database users, i.e. finding specific data, adding records etc. Data definition, transaction control and data control language cover procedures more likely to be used by database developers or administrators, i.e. controlling access etc. [15].

B. Capturing Natural Language Queries

This paper focuses on the type of queries which typical databases users would ask for a specific domain. In order to collect a range of real life natural language requests which might be made of a database, questionnaires were distributed to 10 people who regularly use databases in their main job function. The respondents used a range of database management systems in different domains. However this helped in determining whether particular words or phrases came up regularly, or whether particular query types were common, regardless of database content.

The questionnaire asked some closed questions, to determine the user's job role and the type of queries they made. Open questions were then used to ask for a description of their database, and to give, up to 5 examples of questions/requests, which they might use if they could communicate with the database in English. Users were also asked to provide any alternative phrasing for each question or request. This was to achieve a wider range of possible ways in which different wording could be used. Two aspects of the questionnaire results were evaluated; the language used and the type of resulting SQL query. As the people completing the questionnaires used different databases, only language which referred to database-independent SQL terms was analysed.

The results were used to identify the words which people most commonly use when referring to SQL terms. For example, the use of the words "total", "amount", "how much" etc. indicated that the SQL query would include the term SUM. These findings (summarised in Table I) were used as starting point for creating scripts in the conversational agent.

Table I Summary of Relevant Words Used in Questionnaires

SQL TERM	***Words used in questionnaires***
COUNT	Count, number, how many
SUM	Total, amount, how much
AVERAGE	average
<	under
>	over
GROUP	Breakdown, per, each
DATE VALUES	This week/month etc, last week/month etc, today, actual date, i.e. May 2006

The second point considered was with regards to natural language was the respondents' use of "alternative phrasing". Often the alternative phrasing actually asked a slightly different question. The most common example of this was when respondents asked for both a listing and a count of attributes which met a certain criteria. For example:
"How many organisations are in MAIN?"/"Which organisations are in MAIN?"
This would seem to indicate that the respondent can be flexible about the exact format of the query results, so long as it gives them the information which they need. Some respondents also used the "alternative phrasing" section to add criteria, indicating that they might start with a simple query but then make this more complex and specific.

C. Type of SQL query

The results showed that all the requests were concerned with data retrieval, and could be mapped onto "SELECT" queries. The most common type of queries identified from this study can be categorized into two templates:

a) Limiting selection by using a comparison condition or conditions. E.g.
SELECT att1 (att2 etc)
FROM table(s)
WHERE another_att MEETS CRITERIA (=/</> BETWEEN values)
Examples :
"List of students on a specific course with their addresses and contact details"/"Which organisations give employment advice?"

b) Counting the number of rows/summing the numeric values of rows which meet a comparison condition or conditions
SELECT SUM/COUNT (att1) *att2 etc*
FROM table*(s)*
WHERE another_att MEETS CRITERIA (=/</> BETWEEN values)
GROUP BY att2 etc.
Examples:
"How many students are on a particular course?"/"How much money have we made from courses in the last month?"/How much does it cost the surgery each month for paying prescriptions"

The most common criteria specified were:

- Database attribute with a text data type equals a particular text value.
- Database attribute with a date data type falls between two date values.
- Database attribute with a numeric data type is greater or less than a numeric value.

As only a relatively small range of natural language questions were collected from the questionnaires, further types of natural language questions were collected from the ELF website, where the makers of ELF have tested its performance compared to Microsoft English Query and Linguistic Technology's English Wizard [16]. These included some more complex queries, such as nested SELECT statements (i.e. "What products have sold more than onions"), finding absence of data (i.e. "Who has not bought wine"), and finding records which all meet only one criteria (i.e. "Who has only bought wine").

Data analysed from the questionnaires, and from the ELF website have indicated that a large number of queries can be mapped onto the two SQL templates, (a) and (b). Consequently, this paper will focus on these relatively simple types of SQL SELECT queries, as they provide sufficient coverage of real life data retrieval requests made by database users. There are a number of rules which need to be followed to construct the (a) and (b) SQL templates. Every SQL SELECT query must contain at least one database attribute in its SELECT clause and at least one relation in its FROM clause. If a WHERE clause is present, it must contain at least one database attribute, with a corresponding comparison term (=/</>/BETWEEN) and a text, numeric or date value. If both an aggregated and non-aggregated database attribute(s) are present in the SELECT clause, then the non-aggregated database attribute(s) must also appear in a GROUP BY clause. Finally, if more than one relation is present in the FROM clause, the WHERE clause must contain details of the database attributes on which the tables are joined, i.e.WHERE customers.customer_id = orders.customer_id [17]. The expert system's rule base must ensure that these rules are followed. It can also use these rules to determine where the different elements of user input will fit into an SQL query. For example, if a text value is present in user input, and also present in the database, it will be put into the WHERE clause, along with the database attribute which contains it, and the "=" sign, i.e. WHERE customer_name='Tesco'.

IV. EXPERIMENTAL METHODOLOGY

A. Creating a dataset

It was necessary to create a dataset of natural language questions to be used on the system during development. This was done by determining which of the natural language requests from the questionnaires were simple enough to map onto the two SQL templates which had been identified. These then had to be adapted so that they related to the content of the system's database. Examples of adapted questions and corresponding SQL queries are shown in Table II.

Table II: Sample Natural Language Queries

Type (a)Questions	**SQL**
Which customers are based in France?	SELECT customer_name FROM customers WHERE country='France';
Show all the products which have more than 2000 in stock.	SELECT product_name FROM products WHERE no_in_stock>2000;
How much does instant coffee cost?	SELECT product_price from products WHERE product_name='instant coffee';

Type (b) Questions	SQL
How many large customers do we have?	SELECT COUNT(customer_name) FROM customers WHERE company_type='large';
How many products cost less than £4.00?	SELECT COUNT(product_name) FROM products WHERE product_price<?4.00;
How much steak have we sold?	SELECT SUM (no_of_products), SUM (total_price) FROM products p, order_details od WHERE p.product_name=od.product_name AND product_name='steak';

B. Developing Conversational Agent Scripts

The system has to identify all the relevant attributes of user input, such as database attribute names, database values, numbers, dates, comparative terms such as “more than”, “less than” etc. This will be done by the conversational agent, which will detect and capture key attributes in user input.
A different script was developed to identify the presence or absence of each possible relevant attribute in user input.
The conversational agent was developed to capture the following:

- Any reference to database attributes or values
- Any reference to aggregate terms, such as “how much”, “total”, “sum” etc. It is also necessary to determine which database attribute or value the aggregate term refers to. Analysis of the questionnaires showed that the element being counted or summed is normally the first element which is mentioned after the counted or summed term.
- Any reference to numerical, date or currency values. It is also necessary to determine which comparative term is being used in reference to these values, i.e. more than, less than, or equals. Therefore three different scripts were written for each of the three value types.

Once the conversational agent had identified all the relevant elements of user input, these are passed on, as text, to a class which sets them as string variables, i.e. if the user has referred to customers, a string variable is set to “customer_name” and added to a corresponding Vector; if an aggregate term has been used in relation to a database value (i.e. “How many apples”), then two corresponding string variables are set to the name of the value itself, and the name of the database attribute it belongs to. These variables are then passed on to the expert system, which uses rule based reasoning to build an SQL query from them, as described in Section II.

C. Testing strategies

Testing was done by a number of people who use databases regularly in their major job function. The testers were given a description of the system database, and an explanation of the types of questions which the system could deal with. They were then asked to type a number of questions of their own choice, using their own words, into the conversational agent interface. The query results were then displayed in a new window. At the same time, the users' questions were automatically written to a log file, along with data which demonstrated how different attributes were captured by the conversational agent during the process. After each tester had completed testing, the log file produced by the conversational agent was saved to be analysed later.

V. Results and Discussion

73 questions were asked by users during testing. 24 of these questions were disregarded as they had not been covered in the domain of the prototype system. Of the 49 suitable questions, 42 resulted in a correct SQL query. Analysis of these questions and the queries which were generated revealed the following:

- 30 of the suitable questions processed by the system gave correct results which displayed only relevant information. This included questions with multiple criteria, for example, “What amount of onions did Tesco buy in May”.
- 12 of the suitable questions processed by the system gave correct results, but also displayed information not asked for in the question. The most common example of this was when sales totals were displayed when they were not requested. The conversational agent identifies any words connected with sales (i.e. “sold”, “bought”, “distributed” etc.) and the expert system interprets this as a request for sales totals (both total cost and number of products sold). However, while in some cases, a reference to sales does require sales totals (i.e. How many apples were sold in June?”), other do not, (i.e. “what products are sold by Tesco”). The scope of this prototype system failed to detect this subtle distinction, and so the question “What products are sold by Tesco?” displays the product names and the sales totals for each product.
- 3 questions failed to answer the user’s question. For example one question, “how many products do we distribute”, returned the total number of units sold rather than distribute. In this instance the expert system interpreted this as a request for sales totals. A further question, “which products are sold by Aldi that cost over £2.50”, failed because the database contained more than one database attribute with a currency data type. In this instance further clarification was required with the user through the conversational agent as the expert system could not determine which of these databases attributes the value of £2.50 referred too.
- 4 questions failed because there was not enough information in the question, i.e. “describe my customers”, “What information do we have about vegetables”.

There were a number of testers' questions which were not grammatically correct, i.e. “What large customers are they” and “list all amount of sales in May”. The system dealt with

these questions successfully, as it could still identify specific attributes. Another issue which arose during testing was the ambiguous nature of some of the testers' questions. For example in the question “What was the total amount of orders placed by Spain”, it is unclear whether the user required the total number of orders placed, or the total cost of these orders. Also, the question, “What are the sizes of customer that we deal with” does not make it clear whether the user wants just a list of the different sizes which exist in the database, or whether they want all customer names and their corresponding sizes.

The simple errors identified during the testing phase were due to the limited scope of the scripts written for this specific application and not an inherent failure of the proposed system. The errors can easily be avoided in future prototypes by the development of a complete set of scripts for the conversational agent. In all these instances the benefits of using a conversational agent were obvious in any ambiguity in the users’ natural language request could be rectified through further interaction.

VI. Conclusions and Further Work

This paper has proposed and shown that a conversational agent and expert system can be used as a NLDI. The conversational agent can capture key attributes of user input, such as database attributes and associated values, aggregate terms etc. which the expert system can then transform into an SQL query, using rule based reasoning. The prototype system had a success rate of 85.7% during user testing. 100% accuracy is difficult to achieve in the area of natural language as user input can be ambiguous. However a conversational agent is only as good as the effort which is put into the scripting process. The errors in the prototype system were mainly caused by the limited scope of the scripts that were written. The interactive potential of a conversational agent could be developed further to deal with this problem. When the system returned incorrect results, they were normally close to a correct answer, often simply omitting some required database attributes or returning unnecessary ones. It would be useful if the system could return an explanation of the results, so that the user could check that the system has interpreted their request correctly. A facility could then be included to allow the user to modify the results, i.e. adding or removing a database attribute, if the system has misinterpreted their question. Allowing users to modify inaccurate results in this way would also allow the modification of correct results, for example, if the user wanted to change one element, or add further criteria.

While the results are promising, clearly, more research would need to be done in order to develop this system further. A much larger survey would need to be done with database users, in order to collect a wider range of example natural language questions in order to exploit the interactive capability of the conversational agent. A more interactive system would therefore require extensive user involvement during development.. It would also be beneficial to conduct more research on a range of real life databases, to discover whether databases with different type’s content still share any similarities in their schema, and how this relates to the type of questions which their users ask.

Acknowledgment

The authors wish to thank Convagent Ltd for the use of their conversational agent engine for use within this prototype system.

References

[1] M. Wallace, Communicating with Databases in Natural Language, Ellis Horwood, Chichester, 1984.
[2] R. A. Bhootra, “Natural Language Interfaces: Comparing English Language Front End and English Query”, Master's thesis , Virginia Commonwealth University, 2004.
[3] I. Androutsopoulos, G. Ritchie, P. Thanisch, 'MASQUE/SQL – An Efficient and Portable Natural Language Query Interfaces for Relational Databases', in Proceedings of the 6th International Conference on industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1993.
[4] S.A. Knowles, A Natural Language Database Interface for SQL-Tutor, Honours Project, Nov 1993.
[5] A.Popescu, O. Etzioni, H. Kautz, 'Towards a Theory of Natural Language Interfaces to Databases', in Proceedings of the 8th International Conference on Intelligent User Interfaces, Miami, Florida, 2003, pp 149-157.
[6] Mauldin, M. Chatterbots, Tinymuds, And The Turing Test: Entering The Loebner Prize Competition, AAAI 1994
[7] Colby, K. Artificial Paranoia: A Computer Simulation of Paranoid Process, Pergamon Press., New York, 1975
[8] ALICE (Artificial Linguistic Internet Computer Entity) Available: http://www.alicebot.org.
[9] ADAM, Convagent Ltd, Available: http://www.convagent.com
[10] J. Weizenbaum, "ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine," Communications of the Association for Computing Machinery 9 (1966): 36-45.
[11] D Michie, C Sammut, Infochat Scripter's Manual, Convagent Ltd, Manchester, UK, 2001.
[12] Turing, A. Computing Machinery and Intelligence. Mind 49: 433-460, 1950.
[13] J. Bigus & J. Bigus, Constructing Intelligent Agents Using Java, Wiley, New York, 2001.
[14] S. Cannan, G.Otten, SQL: The Standard Handbook, McGraw-Hill, London, 1993, p8.
[15] C. Ritchie, Relational Database Principles, Continuum, London, 2002, p128.
[16] ELF Software CO ELF Software Documentation Series: Examples, Available http://www.elfsoft.com/ns/help/accelf/Examples.htm
[17] J.Bowman, S. Emerson, M Darnowsky, The Practical SQL Handbook: Using SQL Variants, Addison-Wesley , London, 2001.