A Response to the Need for Summary Responses

J.K. Kalita, M.J. Colbourn+ and G.I. McCalla
Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan, S7N 0W0
CANADA

Abstract

In this paper we argue that natural language interfaces to databases should be able to produce summary responses as well as listing actual data. We describe a system (incorporating a number of heuristics and a knowledge base built on top of the database) that has been developed to generate such summary responses. It is largely domain-independent, has been tested on many examples, and handles a wide variety of situations where summary responses would be useful.

1. Introduction

For over a decade research has been ongoing into the diverse and complex issues involved in developing smart natural language interfaces to database systems. Pioneering front-end systems such as PLANES [15], REQUEST [12], TORUS [11] and RENDEZVOUS [1] experimented with, among other things, various parsing formalisms (e.g. semantic grammars, transformational grammars and augmented transition networks); the need for knowledge representation (e.g. using production systems or semantic networks); and the usefulness of clarification dialogue in disambiguating a user's query.

Recent research has addressed various dialogue issues in order to enhance the elegance of the database interactions. Such research includes attempts to resolve anaphoric references in queries [2,4,14,16], to track the user's focus of attention [2,4,14,16], and to generate cooperative responses. In particular, the CO-OP system [7] is able to analyze presumptions of the user in order to generate appropriate explanations for answers that may mislead the user. Janas [5] takes a similar approach to generate indirect answers instead of providing direct inappropriate ones. Mays [8] has developed techniques to monitor changes in the database and provide relevant information on these changes to the user. McCoy [9] and McKeown [10] attempt to provide answers to questions about the structure of the database rather than extensional information as to its contents. We investigate herein, one particular approach to generating "non-extensional" responses - in particular the generation of "summary" responses.

Generating abstract "summary" responses to users' queries is often preferable to providing enumerative replies. This follows from an important convention of human dialogue that no participant should monopolize the discourse (i.e. "be brief" [3]). Furthermore, extensional responses can occasionally mislead the user where summary responses would not. Consider the following example [13]:

> Q1: Which department managers earn over $40k per year?
> S1-1: Abel, Baker, Charles, Doug.
> S1-2: All of them.

By enumerating managers who earn over $40k, the first response implies that there are managers who *do not* earn that much. In linguistic pragmatics, this is called a *scalar implicature* [3]. In circumstances where the user is liable to infer an invalid scalar implicature, the system should be able to produce an appropriate response to block the generation of such an inference as is done by the response S1-2.

2. Overview of the System

We describe herein a system which has been developed for the generation of summary responses to user's queries (fully detailed in [6]). The system arrives at concise responses by employing a search of the relevant data for the existence of "interesting" patterns. It uses heuristics to guide this search and a knowledge base to enhance efficiency and help determine "interestingness".

The database used to test the system is a simple

+ Now, at the Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, CANADA

relational database of student records, although the methods developed are largely domain-independent. In order to concentrate on the response generation issues, the input/output for the system is in an internal form - an actual parser and surface language generation capabilities will be incorporated in future versions of the system.

The flow of control in the system is simple. The formal representation of the query is used to access the database and obtain the tuples which satisfy the user's query (which we will call T_{qual}; the other tuples will be called T_{unqual}). After the data is accessed, the system, in consultation with its knowledge base, calls upon its heuristics to find interesting non-enumerative patterns. The heuristics are tried in order, until one succeeds or all fail. When a heuristic detects an appropriate pattern, the system terminates the search and produces the response as dictated by the successful heuristic. If all heuristics fail, the system reports its inability to produce a descriptive response. In any event, the user may ask the system to produce an extensional answer by listing the data if he/she so desires.

3. The Heuristics

The heuristics employed in the system are procedural in nature. They guide the system to search for various patterns that may exist in the data. The heuristics are linearly ordered; they range from simple to complex. The ordering of the heuristics assumes that if more than one descriptive answer can be obtained for a query, it is sensible to produce the "simplest" one.

The *equality heuristic* determines if all data values appearing for a particular attribute A in T_{qual} are the same (say, α). If so, and if no tuple in T_{unqual} has the same value for the attribute A, the general formulation of the response is:

"All tuples having the value α for attribute A."

The particular value under consideration must be one of the designated "distinguishing values" for the attribute. Response S1-2 (above) is an example of what this heuristic would do.

The dual of the equality heuristic is the *inequality heuristic* where instead of looking for equalities, the system searches for inequalities. The inequality heuristic enables the system to produce responses such as:

Q2: Which students are taking makeup courses?
S2: All students with non-Computer Science undergraduate background.

Here, the value "Computer Science" for the attribute UNIVERSITY-DEPARTMENT in the database under consideration may be considered a distinguishing value.

If the equality or inequality heuristics are not applicable in their pure form and there are a "few" ("few" depends on the relative number of tuples in T_{qual} and T_{unqual} and some other factors) tuples in T_{unqual} which do not satisfy the requirement of the heuristic, a modification of the response produced by the heuristic may be presented to the user. An example of such a modification is seen in the following:

Q3: Which students are receiving University scholarships?
S3: All but one foreign students. In addition, two Canadian students are also receiving University scholarships.

Another set of heuristics, the *range heuristics*, determine if the data values for an attribute in the tuples in T_{qual} are within a particular well-defined range. There are two main types of range heuristics - one is concerned with maximum values and the other with minimum values. We will discuss only the maximum range heuristic here.

The maximum heuristic determines if the values of an attribute for all tuples in T_{qual} are below a particular limit while the values of the attribute in all tuples in T_{unqual} are not. An example response produced by the maximum heuristic is:

Q4: Which students have been advised to discontinue studies at the University?
S4: All students with a cumulative GPA of 2.0 or less.

In some cases, the maximum and minimum heuristics may be used together to define the end-points of a range of values (for some attribute) which the tuples in T_{qual} satisfy. This results in a range specification. If α is the minimum value and β is the maximum value of the attribute A in T_{qual}, then the corresponding response is:

"All tuples with the value of attribute A ranging from α to β"

An example of an answer with range specification is:

Q5: Which students are in section 1 of CMPT110.3?
S5: All students with surnames starting with 'A' through 'F'.

There are several heuristic rules which the system follows in producing answers with range specification. For example, one of these rules limits the actual range specified in an answer to 75% or less of the potential range of the attribute values. This limitation of 75% is not sacrosanct; it is an arbitrary decision by the implementor of the knowledge base. In the current implementation it is believed that if the actual range is more than 75% of the potential range, no special meaning

can be attributed to the occurrence of this range in T_{qual}.

Another rule requires that the actual range specified in an answer must not be so small as to identify the actual tuples which constitute the answer. For example, we should not produce a response such as:

> "All students with student-id-no between 821661 and 821663"

In fact, such answers are not brief when compared to the size of the set of tuples which they qualify.

A more complex heuristic is the *conjunction heuristic*. If all values of an attribute A in T_{qual} satisfy a relation R (in the mathematical sense) and there are some tuples in T_{unqual} in which the values of the attribute A satisfy this relation R, the system attempts to determine via the above heuristics if there is/are some "interesting" distinguishing characteristic(s) which the set T_{qual} satisfies, but the set of tuples in T_{unqual} satisfying the relation R do not. Let us call the distinguishing characteristic(s) D. The general formulation of the response is

> "All tuples which satisfy the relation R for attribute A and have the characteristic(s) D."

An example is:

> Q6: Which students are working as T.A. and R.A.?
>
> S6: Students who have completed at least two years at the University and who are not employed outside the University.

If none of the above heuristics can be applied successfully, the *disjunction heuristic* attempts to divide the tuples in T_{qual} into a number of subsets and determine whether the above heuristics are appropriate for all of these subsets. The number of such subsets should be "small"; if too many subsets are identified, it is no more elegant than listing the data, which we are trying to avoid. The number of allowable subsets partially depends upon the number of tuples in T_{qual}. An example showing three partitions based on the values of three different attributes is:

> Q7: Which graduate students are not receiving University scholarships?
>
> S7: Students who are receiving NSERC scholarships or have cumulative GPA less than 6.0 or have completed at least two years at the University.

If none of the above heuristics produces a satisfactory response, the *foreign-key heuristic* searches other "related" relations. A related relation is one with which the relation under consideration has some common or join attribute(s). The names of such related relations and the attributes via which such a relation can be joined with the original target relation can be obtained from the knowledge base to be discussed later. An example of such a dialogue is:

> Q8: Which students are taking 880-level courses?
>
> S8: All second year students. In addition, two first year students are also taking 880-level courses.

While attempting to answer Q8, the system finds that the question pertains to the relation COURSE-REGISTRATIONS. However, it fails to obtain any interesting descriptive pattern about the tuples in T_{qual} by considering this relation alone. Hence, the system consults the knowledge base and finds that the relation COURSE-REGISTRATIONS can be joined with the relation STUDENTS. It takes the join of all the tuples constituting T_{qual} with the relation STUDENTS and projects the resulting relation on the attributes of the relation STUDENTS. Let us call these tuples $T_{new\text{-}qual}$. Next, it attempts to discover the existence of some pattern in the tuples in $T_{new\text{-}qual}$. It succeeds in producing the response given in S8 by employing modified equality heuristic.

4. The Knowledge Base

The knowledge base incorporates subjective perceptions of the user as to the nature and contents of the database. It consists of two types of frames - the *relation* and the *attribute* frames. These frames may be considered to be an extension of the database schema. The frames are created by the interface builder, and different sets of frames must be provided for different types of users and/or different databases.

Each *relation frame* corresponds to an actual relation in the database; it provides the possible links with all other relations in the database. In other words, these frames define all permissible joins of two relations. If a direct join is not possible between two specific relations, the frame contains the name of a third relation which must be included in the join. The information in the relation frames is useful in the application of the foreign-key heuristic.

The attribute frames play a role in our system similar to that played by McCoy's axioms [9]. Each *attribute frame* corresponds to an attribute in the relations in the database. In addition to a description of the attributes, these frames indicate the nature and range of the attribute's potential values. The expected range of values that an attribute may assume is helpful to the range heuristics. Information regarding the relative preferability of the various attributes is also included.

Each attribute frame also contains a slot for "distinguishing values" which the attribute might take. This slot provides information for distinguishing a subclass of an entity from other sub-classes. The contents of this field are useful in producing descriptive responses to users' queries. This slot contains one or more clauses, each of the following format ('[ ]' means optionality; '...' means arbitrary number of repetitions of the immediately preceding clause):

(list-of-distinguishing-values-1
(applicable-operator-1-1 [denomination-1-1])
[(applicable-operator-1-2 [denomination-1-2])]
...)

If all the values of the attribute in T_{real} satisfy "applicable-operator-1-1" with respect to the contents of the list "list-of-distinguishing-values-1", the actual values may be termed as "denomination-1-1" for producing responses. If the value of "denomination-1-1" is null, no names can be attached to the actual values of the attribute.

The *Distinguishing Values* slot enables the implementor to specify classifications that he would *a priori* like to appear meaningfully in descriptive responses. This information enables the system to faithfully reflect the implementor perceived notions regarding how a database entity class may be appropriately partitioned into subclasses for generating summary responses.

It is often useful to provide descriptive answers on the basis of certain preferred attributes. For example, for the STUDENTS relation, it is more "meaningful" to provide answers on the basis of the attribute NATIONALITY or UG-MAJOR, rather than STUDENT-ID-NO or AMOUNT-OF-FINANCIAL-AID. However, it is impossible to give a concrete weight regarding each attribute's preferability. Therefore, we have classified the attributes into several groups; all attributes in a group are considered equally useful in producing meaningful qualitative answers to queries.

This classification means that it is preferable and more useful to produce descriptive responses using the attributes in preference category 1 than the attributes in category 2, 3 or 4. This categorization is based on one's familiarity with the data. The decision is subjective, and hence it is bound to vary according to the judgement of the person building the interface. In the *Preference Category* slot, we have an entry corresponding to each relation the attribute occurs in. The information in this slot ensures that the system chooses a description based on the most salient attribute(s) for producing a response.

A simple example of an attribute frame is given below:

Name:- (NATIONALITY, STUDENTS)
Nature-of-Attribute:- String of characters
Distinguishing-Values:-
(((Canadian)(=)($\neq$ foreign))
((U.K. U.S.A. Australia ...)
(*member-of* English-speaking countries))
((U.K. France ...)
(*member-of* Europe))
Potential-Range:- Any member from a given list of countries
Rounding-off-to-be-done?:- Not applicable
Preference-Category:- 1

The example shows the frame for the attribute NATIONALITY belonging to the STUDENTS relation. It assumes character values. To be valid, the values must be members of a previously compiled list of countries. It belongs to the preference category 1 discussed above. Let us consider the clause ((Canadian)(=)($\neq$ foreign)) in the *Distinguishing Value* slot. The value "Canadian" is a distinguishing value in the domain of values which the attribute may take. The term "(=)" indicates that it is possible to identify a class of students using the descriptive expression "NATIONALITY = Canadian". If NATIONALITY $\neq$ "Canadian", the student may be referred to as a "FOREIGN" student. Similarly, if the value for a student under the attribute NATIONALITY is a member of the set (U.K. U.S.A. Australia ...), he may be designated as coming from an English-speaking country. This information may be helpful in answering a query such as:

Q9: Which students are taking the "Intensive English" course in the Fall term?
S9: Most entering foreign students from non-English speaking countries.

5. Concluding Remarks

A system incorporating the details explained above has been implemented and extensive experiments have been performed using a simple student database. Every heuristic has demonstrated its usefulness in producing summary responses by being successful in this environment. The heuristics are domain-independent, and the knowledge base is easily modifiable to adapt to the requirements of a new user or database domain.

For performance enhancement, the knowledge base may be augmented with an additional component for storing away the results of the preceding database interactions to obviate the need to search the database for every query. The extended knowledge base may be utilized for improved modelling of the user's beliefs and perceptions about the data by providing a mechanism

to introduce the user's own definitions and descriptive terminologies. Further research is necessary in order to obtain an acceptable structure for this additional component of the knowledge base. In addition, the factors - linguistic or otherwise, that influence the appropriateness of the generation of a summary response for a give question at a particular point in the interaction are also to be investigated.

Generation of summary responses has important implications if the interactions with a database management system are to have the properties and constraints normally associated with human dialogue. Interactions with traditional database management systems lack the "intelligence" and elegance which we ascribe to human behaviour. We feel that providing summary responses will be an important tool to be used in achieving database interfaces that behave intelligently and co-operatively.

Acknowledgements

We would like to thank the National Science and Engineering Research Council of Canada (NSERC) and the University of Saskatchewan for supporting this research both financially and through the provision of computing facilities. We would also like to express our gratitude to Paul Sorensen and Robin Cohen for their many helpful comments during the course of the research.

References

[1] Codd E.F., R.S. Arnold, J.M. Cadious, C.L. Chang, N. Roussopoulos, *RENDEZVOUS Version 1: An Experimental English Language Query Formulation System for Casual Users of Relational Databases*, Research Report No. RJ2144 (29407), IBM Research Laboratory, San Jose, California, 1978.

[2] Davidson J., "Natural Language Access to Database: User Modelling and Focus", *Proceedings of Canadian Society for Computational Studies of Intelligence*, Saskatoon, May, 1982, 204-211.

[3] Grice H.P., "Logic and Conversation", in P. Cole and J.L. Morgan (eds.), *Syntax and Semantics: Speech Acts*, Vol. 3, Academic Press, New York, 1975, 41-58.

[4] Grosz B.J., "The Representation and the Use of Focus in a System for Understanding Dialogues", *Proc. 5th IJCAI*, Cambridge, 1977, 67-76.

[5] Janas J.M., "How to not Say "NIL" - Improving answers to Failing Queries in Data Base Systems", *Proc. 6th IJCAI*, 1979, Tokyo, 429-434.

[6] Kalita J.K., *Generating Summary Responses to Natural Language Database Queries*, M.Sc. Thesis; also available as TR-9, University of Saskatchewan, Saskatoon, 1984.

[7] Kaplan S.J., "Cooperative Responses from a Portable Natural Language Query System", *Artificial Intelligence*, Vol. 19, No. 2, Oct. 1982, 165-187.

[8] Mays E., S. Lanka, A.K. Joshi and B.L. Webber, "Natural Language Interaction with Dynamic Knowledge Bases: Monitoring as Response", *Proc. 7th IJCAI*, Vancouver, 1981, 61-63.

[9] McCoy K.F., "Augmenting a Database Knowledge Representation for Natural Language Generation", *Proc. 20th Annual Conference of the ACL*, Toronto, Ontario, June, 1982, 121-128.

[10] McKeown K.R., "The TEXT system for Natural Language Generation: An Overview", *Proc. 20th Annual Conference of the ACL*, Toronto, Ontario, June 1982, 113-120.

[11] Mylopoulos J., A. Borgida, P. Cohen, N. Roussopoulos, J. Tsotsos, H. Wing, "TORUS : A Step towards Bridging the Gap between Databases and Casual User", *Information Systems*, Vol. 2, 1976, 49-64.

[12] Plath W.J., "REQUEST : A Natural Language Question Answering System", *IBM Journal of Research and Development*, Vol. 20, July 1976, 326-335.

[13] Reiter R., H. Gallaire, J.J. King, J. Mylopoulos and B.L. Webber, "A Panel on AI and Databases", *Proc. 8th IJCAI*, 1983, Karlsruhe, West Germany, 1199-1206.

[14] Sidner C.L., *Towards A Computational Theory of Definite Anaphora Comprehension in English Discourse*", TR-537, AI Laboratory, MIT, Cambridge, Massachussets, 1979.

[15] Waltz D.L., An English Language Question Answering System for a Large Relational Database, *CACM*, Vol. 21, July, 1978, 526-539.

[16] Webber B. and R. Reiter, "Anaphora and Logical Form: On Formal Meaning Representations for Natural Language", *Proc. 5th IJCAI*, Cambridge, Massachussets, 1977, 121-131.