INTERPRETING NATURAL LANGUAGE DATABASE UPDATES

S. Jerrold Kaplan
Jim Davidson
Computer Science Dept.
Stanford University
Stanford, Ca. 94305

1. Introduction

Although the problem of *querying* a database in natural language has been studied extensively, there has been relatively little work on processing database *updates* expressed in natural language. To interpret update requests, several linguistic issues must be addressed that do not typically pose difficulties when dealing exclusively with queries. This paper briefly examines some of the linguistic problems encountered, and describes an implemented system that performs simple natural language database updates.

The primary difficulty with interpreting natural language updates is that there may be several ways in which a particular update can be performed in the underlying database. Many of these options, while literally correct and semantically meaningful, may correspond to bizarre interpretations of the request. While human speakers would intuitively reject these unusual readings, a computer program may be unable to distinguish them from more appropriate ones. If carried out, they often have undesirable side effects on the database.

For example, a simple request to "Change the teacher of CS345 from Smith to Jones" might be carried out by altering the number of a course that Jones already teaches to be CS345, by changing Smith's name to be Jones, or by modifying a "teaches" link in the database. While all of these may literally carry out the update, they may implicitly cause unanticipated changes such as altering Jones' salary to be Smith's.

Our approach to this problem is to generate a limited set of "candidate" updates, rank them according to a set of domain-independent heuristics that reflect general properties of "reasonable" updates, and either perform the update or present the highest ranked options to the user for selection.

This process may be guided by various linguistic considerations, such as the difference between "transparent" and "opaque" readings of the user's request, and the interpretation of counterfactual conditionals.

Our goal is a system that will process natural language updates, explaining problems or options to the user in terms that s/he can understand, and effecting the changes to the underlying database with the minimal disruption of other views. At this time, a pilot implementation is complete.

2. Generating Candidate Updates

Before an appropriate change can be made to a database in response to a natural language request, it is useful to generate a set of "candidate" updates that can then be evaluated for plausibility. In most cases, an infinite number of changes to the database are possible that would literally carry out the request (mainly by creating and inserting "dummy" values and links). However, this process can be simplified by generating only candidate updates that can be directly derived from the user's phrasing of the request. This limitation is justified by observing that most reasonable updates correspond to different readings of expressions in *referentially opaque* contexts.

A referentially opaque context is one in which two expressions that refer to the same real world concept cannot be interchanged in the context without changing the meaning of the utterance [Quine, 1971]. Natural language database updates often contain opaque contexts.

For example, consider that a particular individual (in a suitable database) may be referred to as "Dr. Smith", "the instructor of CS100", "the youngest assistant professor", or "the occupant of Rm. 424". While each of these expressions may identify the same database record (i.e. they have the same *extension*), they suggest different methods for locating that record (their *intensions* differ). In the context of a database query, where the goal is to unambiguously specify the response set (extension), the method by which they are accessed (the intension) does not normally affect the response (for a counterexample, however, see [Nash-Webber, 1976]). Updates, on the other hand, are often sensitive to the substitution of extensionally equivalent referring expressions. "Change the instructor of CS100 to Dr. Jones." may not be equivalent to "Change the youngest assistant professor to Dr. Jones." or "Change Dr. Smith to Dr. Jones." Each of these may imply different updates to the underlying database.

This characteristic of natural language updates suggests that the generation of candidate updates can be performed as a *language driven inference* [Kaplan, 1978] without severely limiting the class of updates to be examined. "Language driven inference" is a style of natural language processing where the inferencing process is driven (and hence limited) by the phrasing of the user's request. Two specific characteristics of language driven inference are applied here to control the generation process.

First, it is assumed that the underlying database update must be a series of transactions of the same *type* indicated in the request. That is, if the update requests a deletion, this can only be mapped into a series of deletions in the database. Second, the only kinds of database records that can be changed are those that have been mentioned in some form in the actual request, or occur on paths linking such records. In observing these restrictions, the program will generate mainly updates that correspond to different readings of potentially opaque references in the original request.

3. Selecting Appropriate Updates

At first examination, it would seem to be necessary to incorporate a semantic model of the domain to select an appropriate update from the candidate updates. While this approach would surely be effective, the overhead required to encode, store, and process this knowledge for each individual database may be prohibitive in practical applications. What is needed is a general set of heuristics that will select an appropriate update in a reasonable majority of cases, without specific knowledge of the domain.

The heuristics that are applied to rank the candidate updates are based on the idea that the most appropriate one is likely to cause the minimum number of *side effects* to the user's conception of the database. This concept is developed formally in the work of Lewis, presented in his book on *Counterfactuals* [Lewis, 1973]. In this work, Lewis examines the meaning and formal representation of such statements as "If kangaroos had no tails, they would topple over." (P.8) He argues that to evaluate the correctness of this statement (and similar counterfactual conditionals) it is necessary to construct in one's mind the possible world minimally different from the real world that could potentially contain the conditional (the "nearest" consistent world). He points out that this hypothetical world does not differ only in that kangaroos don't have tails, but also reflects other changes required to make that world plausible. Thus he rejects the idea that in the hypothetical world kangaroos might use crutches (as not being minimally different), or that they might leave the same tracks is the sand (as being inconsistent).

The application of this work to processing natural language database updates is to regard each transaction as presenting a "counterfactual" state of the world, and request that the "nearest" reasonable world in which the counterfactual is true be brought about. (For example, the request "Change the teacher of CS345 from Smith to Jones." might correspond to the counterfactual "If Jones taught CS345 instead of Smith, how would the database be different?" along with a speech act requesting that the database be put in this new state.) To select this nearest world, the number and type of side effects are evaluated for each candidate update, and they are ranked accordingly. Side effects that disrupt the user's view--taken to be the subset of the database that has been accessed in previous transactions--are considered more "severe" than changes to portions of the database not in that view. In data processing terms, the update with the fewest side effects on the user's data sub-model is selected as the most appropriate.

Updates that violate syntactic or semantic constraints implicit in the database structure and content can be eliminated as inconsistent. Functional dependencies, where one attribute uniquely determines another, are useful semantic filters (as in the formal update work of [Dayal, 1979]). When richer semantic data models are available, such as the Structural Model of [Wiederhold and El-Masri, 1979], more sophisticated constraints can be applied. (The current implementation does not make use of any such constraints.)

While this approach can certainly fail in cases where complex domain semantics rule out the "simplest" change--the one with the fewest side effects to the user's view--in the majority of cases it is sufficient to select a reasonable update from among the various possibilities.

4. An Example

The following simple example of this technique illustrates the usefulness of the proposed approach in practical databases. It is drawn from the current pilot implementation.

The program is written in Interlisp [Teitelman, 1978], and runs on a DEC KL-10 under Tenex. An update expressed in a simple natural language subset is parsed by a semantic grammar using the LIFER system [Hendrix, 1977]. Its output is a special version of the SODA relational language [Moore, 1979] that has been modified by Jim Davidson to include the standard database update operations "delete", "insert" and "replace". The parsed request is then passed to a routine that generates the candidate updates, subject to the constraints outlined above. This list is then evaluated and ranked as described in the previous section. If no updates are possible, the user is alerted to this fact. If one alternative is superior, it is carried out. If several updates remain which cannot be compared, they are presented for selection in terms of the effects they will have on the user's view of the database. If the update ultimately performed has unanticipated effects on the user's view (i.e. if the answer to a previous query is now altered), the user is informed.

The example below concerns a small database of information about employees, managers and departments. It is assumed that the user view of the world contains employees and managers, but that s/he does not necessarily know about departments. In the database, managers manage employees "transitively", by managing the departments in which the employees work. For purposes of presentation, intermediate results are displayed here to illustrate the program's actions. Normally, such information would not be printed. Commentary is enclosed in brackets("[ ]").

[Here is a tabular display of the database.]

```
   TABLE DM

DEPT     MGR
---------------
INVNTRY FISHER
MKTING  BAKER
SALES   JONES

  TABLE ED

EMP   DEPT
--------------
ADAMS SALES
WHITE MKTING
BROWN SALES
SMITH INVNTRY
```

[First, the user enters the following query, from which the program infers the user's view of the world.]

```
Enter next command:
(LIST THE EMPLOYEES AND THEIR MANAGERS)

EMP       MGR
--------------------
ADAMS     JONES
WHITE     BAKER
BROWN     JONES
SMITH     FISHER
```

[Next, the user enters a natural language update request.]

```
Enter next command:
(CHANGE BROWN'S MANAGER FROM JONES TO BAKER]
```

[The program now generates the candidate updates. One of these corresponds to moving Brown from the Sales department to the Marketing department. The other would make Baker the manager of the Sales department.]

```
The possible ways of performing the update:

1.   In the relation ED change the DEPT attr of
     the tuple

EMP        DEPT
--------------------
BROWN      SALES

to the value MKTING
```

```
2.   In the relation DM change the MGR attr of
     the tuple

     DPT        MGR
     --------------------
     SALES      JONES

     to the value BAKER
```

[The side effects of each on the user's view are computed.]

```
     These translations have the following side effects
     on the view:

     1. Side effects are:
     Deletions: NIL
     Insertions: NIL
     Replacements: NIL

     2. Side effects are:
     Deletions: NIL
     Insertions: NIL
     Replacements: (ADAMS JONES) -> (ADAMS BAKER)
```

[The program concludes that update (1) is superior to (2), since (2) has the additional side effect of changing Adams' manager to Baker as well.]

```
     Desired translation is: 1.

     Rev'sed view is:

     EMP        MGR
     --------------------
     ADAMS      JONES
     WHITE      BAKER
     BROWN      BAKER
     SMITH      FISHER
```

5. Conclusions

Carrying out a database update request expressed in natural language requires that an intelligent decision be made as to how the update should be accomplished. Correctly identifying "reasonable" resultant states of the database, and selecting a best one among these, may involve world knowledge, domain knowledge, the user's goals and view of the database, and the previous discourse. In short, it is a typical problem in computational linguistics.

Most of the complications derive from the fact that the user has a view of the database that may be a simplification, subset, or transformation of the actual database structure and content. Consequently, there may be multiple ways of carrying out the update on the underlying database (or no ways at all), which are transparent to the user. While most or all of these changes to the underlying database may literally fulfill the user's request, they may have unanticipated or undesirable side-effects on the database or the user's view.

We have developed an approach to this problem that uses domain-independent heuristics to rank a set of candidate updates generated from the original request. A reasonable course of action can then be selected, and carried out. This may involve informing the user that the update is ill-advised (if it cannot be carried out), presenting incomparable alternatives to the user for selection, or simply performing one of the possible updates. Our technique is motivated by linguistic observations about the nature of update requests. Specifically, the use of referential opacity, and the interpretation of counterfactual conditionals, play a role in our design.

A primary advantage of our approach is that it does not require special knowledge about the domain, except that which is implicit in the structure and content of the database. A simple but adequate model of the user's view of the database is derived by tracking the previous dialog, and the heuristics are based on general principles about the nature of possible worlds, and so can be applied to any domain. Consequently, the approach is practical in the sense that it can be transported to new databases without modification.

In part because of its generality, there is a definite risk that the technique will make inappropriate actions or fail to notice preferable options. A more knowledge-based approach would likely yield more accurate and sophisticated results. The process of responding appropriately to updates could be improved by taking advantage of domain specific knowledge external to the database, using partial case-structure semantics, or tracking dialog focus, to name a few. In addition, better heuristics for ranking candidate updates would be likely to enhance performance.

At present, we are developing a formal characterization of the process of performing updates to views. We hope that this will provide us with a tool to improve our understanding of both the problem and the approach we have taken. While the heuristics used in the process are motivated by intuition, there is no obvious reason to assume that they are either optimal or complete. A more formal analysis of the problem may provide a basis for relating the various heuristics and suggest additional ranking criteria.

6. Bibliography

Dayal, U.: *Mapping Problems in Database Systems*, TR-11-79, Center for Research in Computing Technology, Harvard University, 1979.

Hendrix, G.: Human Engineering for Applied Natural Language Processing, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977, 183-191.

Kaplan, S. J.: Indirect Responses to Loaded Questions, *Proceedings of the Second Workshop on Theoretical Issues in Natural Language Processing*, Urbana-Champaign, IL, July, 1978.

Lewis, D.: *Counterfactuals*, Harvard University Press, Cambridge, MA, 1973.

Moore, R.: *Handling Complex Queries in a Distributed Data Base*, TN-170, AI Center, SRI International, October, 1979.

Nash-Webber, B.: *Semantic Interpretation Revisited*, BBN report #3335, Bolt, Beranek, and Newman, Cambridge, MA, 1976.

Quine, W.V.O.: Reference and Modality, in *Reference and Modality*, Leonard Linsky, Ed., Oxford, Oxford University Press, 1971.

Teitelman, W.: *Interlisp Reference Manual*, Xerox PARC, Palo Alto, 1978.

Wiederhold, G. and R. El-Masri: The Structural Model for Database Design, *Proceedings of the International Conference on Entity-Relationship Approach to Systems Analysis and Design*, North Holland Press, 1979, pp 247-267.