Technical Correspondence

A Note on the Utility of Computing Inferences in a Real Data Base Query Environment

It is quite clear that as computer systems are made available to a wider class of more naive users than at present, the interfaces to these systems will need to have properties which have been termed "cooperative", "user friendly" and the like. One of the many desirable features of such an interface is the ability to take into account not only a user's direct input, but also things which are implied in that input, to draw inferences about knowledge assumed but not expressed. A good deal of experimental and theoretical work has gone into determining how to make inferences , what inferences to make, how to stop making inferences, etc. For recent examples, see Charniak (1978), Minsky (1975), Schank and Abelson (1977).

One class of inferences which appear to be rather more computationally tractable than the others are those based on properties of language, i.e., presuppositions and entailments (Weischedel, 1975). These inferences have been studied in the context of natural language query systems in Kaplan (1978), Kaplan and Joshi (1978), and Kaplan (1979). Unlike inferences based on extra-linguistic context, this class is definite, not problematic, and does not lead to unending chains of new inferences. Consider the following example from Kaplan and Joshi (1978):

> "Which programmers in Administration work on project 6471?"

A literal answer of "None" is not nearly as helpful as an answer like:

> "There are no programmers in Administration."

assuming that that is correct with respect to the data base. The papers cited here describe the algorithms used to compute the set of presuppositions and presumptions of queries, and the method of choosing an appropriate response to a question which fails.

Although such a feature might be useful in some query systems, there is an associated cost, both for computing the inferences and for probing the data base to derive a cooperative response. It is of some interest, therefore, to ask how often such a capability would have been useful in a query system operating in a realistic environment rather than in a demonstration situation. Damerau (1981) summarizes experience with the Transformational Question Answering System (TQA) during the first full year of operation, 1978. The full set of 788 queries as listed in that paper were scanned to determine

1. which queries contained presumptions or presuppositions, and
2. how many of these failed to find an answer in the data base.

Of the total query set, 257, or approximately 1/3, contained presuppositions of the kind discussed by Kaplan and Joshi. Of the 61 queries to which the system replied "Nothing in the data base", only 11 contained presumptions and presuppositions of the class being discussed. To be realistic, however, these raw totals should be adjusted to take into account the fact that a number of the presuppositions having to do with the existence of objects will turn out to be true essentially always, (i.e., the users of the data base are not likely to be wrong about the existence of a street or a neighborhood association), although they might well be mistaken about the existence of an address or the name of a property owner. When cases of the former kind are eliminated, only 61 queries of the total, or less than 10 percent, turn out to have sets of presuppositions and presumptions which might reasonably be in error, and only 6 cases of the "Nothing in the data base" type turn out this way. *Thus, in only 6 cases, or less than 1 percent of the queries, would a system which computed presuppositions and presumptions for questions whose answer set was empty have been able to give a better answer than the 1978 TQA system.*[1]

In view of the cost of constructing and integrating a module to handle language related inferences, and in view of the cost of extra data base accesses to derive cooperative answers, it appears that at least in applications like that handled by the TQA system, time and effort would be more profitably spent on other system improvements which would enable the system to handle a wider class of English inputs. Nevertheless, the undoubted theoretical interest and importance of knowing how to deal computationally with inferences

[1] Warren Plath (personal communication) has pointed out that in some cases, presuppositions may have been violated even when the answer set is not empty. For example, the query "In what zone are parcels 5 and 6 in ward 1, block 1?" might return an answer *R5* if, say, parcel 5 was in the data base but parcel 6 was not. In this case, since the answer set is not empty, Kaplan's algorithm would not be invoked, and the user might erroneously conclude that both parcels are in the R5 zone. However, the additional computational load of checking presuppositions individually for every question, not just those which fail, would certainly increase computation time significantly.

remains high, and there is no doubt that a truly natural system for language understanding requires such a capability.

Fred J. Damerau
Mathematical Sciences Department
IBM Thomas J. Watson Research Center
Post Office Box 218
Yorktown Heights, New York 10598

References

Charniak, Eugene (1978). With a Spoon in Hand, This Must be the Eating Frame. In *Theoretical Issues in Natural Language Processing-2, Proceedings*, University of Illinois at Urbana-Champaign, July 1978, pp. 187-198.

Damerau, Fred J. (1981). Operating Statistics for the Transformational Question Answering System. *Am. J. of Comp. Ling.*, this issue, pp. 30-42.

Kaplan, Jerrold S. (1978). Indirect Responses to Loaded Questions. In *Theoretical Issues in Natural Language Processing-2, Proceedings*, University of Illinois at Urbana-Champaign, July 1978, pp. 202-209.

Kaplan, S. Jerrold and Joshi, Aravind K. (1978). Cooperative Responses: An Application of Discourse Inference to Data Base Query Systems. In *Second Annual Conference of the Canadian Society for Computational Studies of Intelligence, Proceedings*, Toronto, Ontario, July 1978.

Kaplan, Jerrold S. (1979). Cooperative Responses from a Portable Natural Language Database Query System. Ph.D. Thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Minsky, Marvin (1975). A Framework for Representing Knowledge. *The Psychology of Computer Vision*, P.H. Winston, ed. New York, McGraw-Hill, pp. 211-277.

Schank, Roger C. and Abelson, Richard P. (1977). *Scripts, Plans, Goals, and Understanding*. Lawrence Erlbaum Press, Hillsdale, New Jersey.

Weischedel, Ralph M. (1975). Computation of a Unique Subclass of Inferences: Presupposition and Entailment. Ph.D. Thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Comments on the Note by Fred J. Damerau

We think the note by Damerau is reasonable, though the discussion is based on a particular system in a particular setting, and so caution is advisable in extrapolating from his observations. However, we would like to make a few comments. (A more detailed response would, of course, require us to examine the 788 queries.) In general, we think there is some confusion in the note regarding the phenomenon of presumption in natural language vs. the applicability of the specific algorithm implemented in the CO-OP system to TQA. The implication that the presupposition analysis is of very limited use in natural language database query systems is, we think, not really properly argued or supported in this note, as elaborated below.

1. It is important to separate the theoretical analysis from the particular implementations and techniques he is discussing. The ratio of responses that may benefit from presupposition analysis is one thing; the difficulty of implementing a particular technique in the TQA system, and whether this is where efforts are best applied is an entirely separate question.

2. If the system is designed appropriately from the beginning, the "additional" cost of performing the presupposition analysis may be minimal or nonexistent. In CO-OP, no additional cost was invoked until an answer was null, i.e. in less than 10 percent of their cases (61 queries). When CO-OP did provide this analysis, it did so in what was in a sense the most straightforward way possible. By contrast, a form of presupposition checking added to the LADDER system (Artificial Intelligence Center of SRI International) was so efficient that it was performed in every case, (as a by-product of answering the query) whether or not required, without any substantive difference in performance.

3. We think the relevant number in the note is that 1/3 of the queries contained checkable presumptions that could be computed via language driven inferences, i.e. from the surface structure of the query. We find this at least plausible, though it would be helpful if the note gave a few examples of queries that were not considered to presume anything. The other figures — how many resulted in empty responses, and how many of these intersected the first class — is highly dependent on the database, user group, etc. In the presented case, the capability of providing indirect responses would have been of use in at most 1 percent of the queries. This suggests that the users had a thorough understanding of the structure and content of the database: in this case the flexibility provided by the natural language interface becomes less significant.

4. Damerau implicitly assumes that it is okay for the system to make misleading responses on occasion. This can be entirely unacceptable to some specific user communities, regardless of how infrequently it occurs — not all queries are born equal. Consider "Do any of the parcels in the unincorporated county owned by large businesses pay property taxes?" The system might answer "no", when, in fact, no such parcels exist. Spurious responses of this type can have disastrous effects — lawsuits, investigations, outraged citizens, etc.

5. Perhaps it is worth pointing out that the ideas in the CO-OP system, if not the specific implementation, have been added successfully to several other query systems. Presumption checking techniques have been incorporated into LADDER, INTELLECT (formerly ROBOT - the Artificial Intelligence Corp.), and REL (Thompson of CALTECH), for example. In general,

these implementations were significantly more efficient, but checked a somewhat narrower class of presumptions than CO-OP.

6. Damerau mentions that queries with non-empty responses can also make presumptions. This is certainly true, even in more subtle ways than noted. (For example, "What is the youngest assistant professors salary?" presumes that there is more than one assistant professor.) Issues such as these are indeed currently under investigation.

Overall, we are pleased to see that Damerau has raised some very important issues and we hope that this exchange will be helpful to the natural language processing community.

Aravind K. Joshi
Dept. of Computer and
Information Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104

S. Jerrold Kaplan
Computer Science Department
Stanford University
Stanford, California 94305

Reply to Joshi and Kaplan

In general, there is little to disagree with in Joshi and Kaplan's comments, but perhaps a couple of points could be clarified.

As Joshi and Kaplan suspected (point 3), the users of this system did indeed thoroughly understand the data base. This makes quite a difference in thinking about the relative importance of facilities in a natural language query system. In particular, such users tend to check strange answers, so that a reply of "no", as in their point 4, would probably result in an additional question of "How many parcels ... ".

With regard to their remarks on implementations that incur no additional cost (points 2 and 5), I would be interested in seeing how presupposition analysis can be done without extra data base retrievals. It would seem that the system would either have to make special retrievals at marked times, as in CO-OP, or would have to make the relevant retrievals for every question so as to have the results available when needed. However, even if the execution time increase were to be zero, we still have a great many other things which we would like to add to our system before we add inference checking.

Fred J. Damerau

Book Reviews

Logic For Problem Solving

Robert Kowalski

Elsevier North Holland, New York, 1979,
287 pp., Paperback, $9.95, ISBN 0-444-00365-7.

This is a textbook introduction to logic programming. Logic programming is based on the premise that programming a task on a computer should begin with a precise formulation of what is wanted. This formulation defines the task clearly; it serves as a theory of the task which can be studied for its implications and limitations. Usually this formulation is computationally inefficient if implemented straightforwardly, but it can be reformulated so that it becomes an efficient program when interpreted by a theorem prover. In this form the logic program is closer to the theory than a PASCAL or LISP program would be, making it easier to verify its correctness and also easier to understand directly.

Logic programming has been applied mostly to formal software specifications, data base systems and problem solving, but it is being applied increasingly to natural language understanding systems [1,2,4,5,6]. In these systems axioms specify the relationship between the input text and whatever representation it is to be parsed into, and between this and whatever the output is to be (e.g., an updated database or the answer to a question). Since these axioms specify the relation between the text and its representation, they form a grammar for the text language, and, as such, are comparable to the rules in a linguist's grammar. When interpreted by a suitable theorem prover, such as a version of PROLOG, they can transform a text into its representation (and often a representation into a text) with practical efficiency.

With logic programming the computational linguist may be able to develop theories of language that are both conceptually well-organized and practical to compute, but this book includes only the most elementary introduction to natural language processing. It uses parsing as an example to show that problems can be solved in ways that correspond to top-down parsing, bottom-up parsing, or an arbitrary mixture of the two, all depending on how the theorem prover decides to