On some arguable claims in B. Shneiderman's evaluation of natural language interaction with database systems

Neil C. Rowe
Department of Computer Science
Stanford University
Stanford, CA 94305

This work is part of the Knowledge Base Management Systems Project, under contract #N00039-80-G-0132 from the Defense Advanced Research Projects Agency of the United States Department of Defense. The views and conclusions contained in this document are those of the author and should not be interpreted as representative of the official policies of DARPA or the US Government.

Introduction

A number of arguable assumptions are made in Ben Shneiderman's article, "A Note on Human Factors Issues of Natural Language Interaction with Database Systems" (*Information Systems, 6*, 2, 125-129, 1981). They include:

1. Computers are precise.

2. Computers *should be* precise.

3. Natural language is imprecise.

4. Formal query languages are precise.

5. Queries a database can't answer are a waste of a user's time.

6. Queries a user has trouble formulating are a waste of a user's time.

7. Natural language "bypasses" speed, storage, and accuracy of computers.

8. Cost-benefit analysis of a query language can prove something.

We dispute these in order.

1. Computers are precise.

Computers are no more necessarily precise than they are necessarily numerical, despite the fact their machine instructions include precise numerical operations. When a system is sufficiently complex its components become obscured. Computers "can be instructed to do exactly what you want without argument or misunderstanding" [3] only to the extent that one writes simple programs for them. The more complex the program task, the more the possibility of unforeseen results. *That*, in fact, is the "greatest advantage of computers" -- they can tell us things we didn't know already, and we didn't precisely say we wanted.

In addition, computers have to be viewed in the context of the information system in which they are used. Many of these uses are quite ill-defined. For instance, a sales manager uses a database of sales statistics to plan an advertising campaign. The information needs are not clear, and it is unfair to call the computer "precise" even though it is supplying "hard facts" -- it's an exploratory tool.

2. Computers *should be* precise.

Computers are symbol manipulators that can be used for a variety of tasks. Whether these symbol manipulations should lead to a precise conclusion, or whether in fact the overall objective itself is clear, is very much up to the user. Information retrieval systems, for instance, use database technology to support very fuzzy user needs. However, information retrieval remains somewhat isolated from the rest of database research and development; the self-perpetuating myth of the "preciseness" of computers attracts a certain personality of people who are not then interested in developing anything besides "precise" applications.

Being precise reflects wanting to control one's environment carefully. This is partly a political issue. Some argue that Western technology is too manipulative, and ought to take more cognizance of existing systems without trying to change them.

In addition, many members of the general public have an antipathy to preciseness. If these people are going to use computers, as indeed technological developments make practically force nearly everyone to do so, we must be prepared to meet their needs with other kinds of interfaces.

3. Natural language is imprecise.

Natural language evolved in order to meet human communication needs as efficiently as possible, within the limits of human information processing capability. It is unlikely that artificially constructed languages, no matter how carefully designed, will begin to approach the subtleties and total information content of natural language, at least for talking about familiar human concepts (cf. [1]). Natural language is just too complicated. Though it sometimes *seems* ambiguous and redundant, this just illustrates our methodological difficulties in grappling with its complexities, not any inherent ambiguity (for people seem to understand one another pretty well most of the time). As we develop the tools through linguistic analysis, language starts to appear more and more regular and rule-based.

Of course for a simple task, a formal language can be more precise than natural language. One can trade off precision and expressiveness. But simple tasks are not the only ones we should consider with databases. Databases are a powerful general information technology that society can use as it sees fit, not just for a few bookkeeping chores in large bureaucracies. As the technology develops, more and more complicated tasks are being supported.

4. Formal query languages are precise.

In a certain sense, artificial languages can be *less* precise than natural languages. Far from "learning the semantics of question asking", users are forced into only one particular mode of thinking about a database and retrieving information from it that may be very narrow (e.g., notions that only certain entities are relations, that only certain entities are attributes, that the database is verbal information only). The friction between the syntactic strait jacket and actual user needs may lead to strange querying errors and inefficiencies. Thus users may end up using artificial languages in highly imprecise ways vis-a-vis their needs, even though their queries may be syntactically precise. It may be like using a precision miniature screwdriver to wedge open a bottle.

5. Queries a database can't answer are a waste of a user's time.

A user issues a database query in order to learn something. If what he learns covers database structure as well as actual data values, so much the better. Any adequate natural language query system should be prepared to discuss structure as well as contents -- users don't draw a very fine distinction. This means a rich conceptual schema plus other related information must be accessible to the interface. Work such as [2, 5, 6] represents a start in using such knowledge.

Just as with a human expert, some of the questions a naive user asks a database may not really make sense. A good expert will try to draw out the germ of the idea, correcting misconceptions where appropriate, not dismissing the questions as a priori nonsensical. Bugs are interesting (cf. [8]). If a user thinks the database can advise him on general management policies (e.g. Shneiderman's example, "Are the managers lenient concerning tardiness and absences?"), he will signal this by queries on policy entities, identifiable from a lexicon lookup. Under parse failure, a helpful interface can try to isolate the "deepest violated precondition", extending [4].

6. Queries a user has trouble formulating are a waste of a user's time.

Natural language is such a rich resource that we can usually say if a user has trouble phasing a question in it, he can't be clear in his own mind what he is asking. The exercise of attempting to formulate may then be educational.

Queries that a natural language interface can't handle because of deficiencies in its linguistic coverage might be considered a kind of query that can't be "formulated". (But users can "formulate" them fine, it's just the system can't *understand.*) Users will be disturbed by these deficiencies, because they can't learn much from them. But this just argues for a comprehensive natural language interface where nearly anything the user can formulate regarding the subject domain is understood, if not acted upon.

7. Natural language "bypasses" speed, storage, and accuracy of computers.

Database applications entail large time delays in accessing often large numbers of records on secondary storage. These time delays make databases a prime candidate for natural language interfaces, because the parsing and linguistic analysis times are likely to be swamped by time for actual data retrieval, even for complex such interfaces. In addition, the size of a database almost certainly swamps the size of the interface code.

It is not true necessarily that natural language querying is more prone to errors than formal query language querying. Since a natural language interface attempts to *do* more, it can get dumped with a larger share of the blame when things go wrong. The same errors made with a formal query language will be blamed instead on the querier's "semantics" or "syntax". Intelligent feedback and verification methods can be supported by a natural language approach as an integral part of the system, decreasing the error rate markedly.

8. Cost-benefit analysis of a query language can prove something.

There are various incompatible ways of measuring costs of a query language [9], but hardly any agreement or insight on the measurement of benefits (cf. [12], ch. 5; [10]; [11]). Until this can be done, analysis of costs is moot for comparing interfaces with even slightly different benefits, which includes most cases.

For instance, error frequency must be measured against language power -- the more extensive the coverage of a language, the more possibilities for error. Just because a tool is more powerful and so can be misused in more ways does not make it less desirable. Comprehensive natural language interfaces for database systems may have such important benefits for some user classes (e.g., computer-afraid users; child users) as to make error costs on retrieval-only usage irrelevant. And busy management cannot be bothered to remember any details of *any* artificial language, or to use time-wasting menu navigation. In fact, since developments enhancing query capabilities like natural language will tend to shift the user population upwards in the management hierarchy ([7]), the notion of the need for rigidly defined criteria in task performance that is so important in middle-level management may become mostly obsolete, making previous studies irrelevant. So new upper-level management users will have rather different kinds of information needs from nearly all previous users, which probably cannot be measured by similar standards -- assuming we did have a clear set of current standards, which we do not.

In addition, there are cases where a cooperative database interface for naive users is perhaps morally necessary regardless of cost. It is within the rights of the citizens of the United States to demand full access to all unclassified and unsensitive government records, beyond the summary information that is published; the Constitution could be so construed. It may be argued the government has the obligation to provide terminals

and cooperative interfaces to such records, in the same way that an elected official is obliged to provide certain basic services for constituents.

A final note

Generally speaking, the thrust of Shneiderman's paper is correct: so-called "natural language" query interfaces, in the various forms available with the current state of the art or in the near future, are not appropriate for nearly all existing databases, as judged by conventional cost-benefit analysis as far as it can be applied. This is because good such interfaces are hard, and much work remains to be done. However, the arguments and evidence amassed in Shneiderman's paper are unconvincing and rely on a number of controversial hidden assumptions. These assumptions reflect a rather narrow class of database users and usage which may not very well characterize the future, or perhaps the future we should encourage.

Acknowledgments

Thanks to Jim Davidson and Kathleen Gilbert.

References

[1] Margaret Boden.
Artificial Intelligence and Natural Man.
Basic Books, New York, 1977.

[2] Tim Finin, Bradley Goodman, and Harry Tennant.
JETS: Achieving Completeness Through Coverage and Closure.
In *Proceedings of Sixth Conference*, pages 275-281. International Joint Conference on Artificial Intelligence, Tokyo, Japan, August, 1979.

[3] I. D. Hill.
Wouldn't It Be Nice If We Could Write Computer Programs In English -- Or Would It?
Honeywell Computer Journal 6(2):76-83, 1972.

[4] S. Jerrold Kaplan.
Cooperative Responses from a Portable Natural Language Database Query System.
Technical Report HPP-79-19, Stanford University Heuristic Programming Project, Stanford, CA, July, 1979.

[5] Eric Mays.
Correcting Misconceptions about Database Structure.
In *Proceedings of Biennial Conference*, pages 123-128. Canadian Society for Computational Studies of Intelligence, Victoria BC, May, 1980.

[6] Kathleen R. McKeown.
Generating Relevant Explanations: Nautral Language Responses to Questions About Database Structure.
In *Proceedings of the First Biennial Conference*, pages 306-309. American Association for Artificial Intelligence, Stanford CA, August, 1980.

[7] Abbe Mowshowitz.
The Conquest of Will: Information Processing in Human Affairs.
Addison-Wesley, Reading MA, 1976.

[8] Seymour Papert.
Mindstorms: Children, Computers, and Powerful Ideas.
Basic Books, New York, 1980.

[9] Phyllis Reisner.
Human Factors Studies of Database Query Languages.
ACM Computing Surveys 13(1):13-31, March, 1981.

[10] Harry Tennant.
What Makes Evaluation Hard?
In *Proceedings of the 19th Annual Meeting*, pages 37-38. Association for Computational Lingustics, Stanford CA, June, 1981.

[11] Bozena Henisz Thompson.
Evaluation of Natural Language Interfaces to Database Systems.
In *Proceedings of the 19th Annual Meeting*, pages 37-38. Association for Computational Lingustics, Stanford CA, June, 1981.

[12] Gio Wiederhold.
Database Design.
McGraw-Hill, New York, 1977.