Parsing in the Absence of a Complete Lexicon

Jim Davidson and S. Jerrold Kaplan
Computer Science Department, Stanford University
Stanford, CA 94305

I. Introduction

It is impractical for natural language parsers which serve as front ends to large or changing databases to maintain a complete in-core lexicon of words and meanings. This note discusses a practical approach to using alternative sources of lexical knowledge by postponing word categorization decisions until the parse is complete, and resolving remaining lexical ambiguities using a variety of information available at that time.

II. The Problem

A natural language parser working with a database query system (e.g., *PLANES* [Waltz et al, 1976], *LADDER* [Hendrix, 1977], *ROBOT* [Harris, 1977], *CO-OP* [Kaplan, 1979]) encounters lexical difficulties not present in simpler applications. In particular, the description of the domain of discourse may be quite large (millions of words), and varies as the underlying database changes. This precludes reliance upon an explicit, fixed *lexicon*—a dictionary which records all the terms known to the system—because of:
(a) *redundancy*: Keeping the same information in two places (the lexicon and the database) leads to problems of integrity. Updating is more difficult if it must occur simultaneously in two places.
(b) *size*: A database of, say, 30,000 entries cannot be duplicated in primary memory.

For example, it may be impractical for a system dealing with a database of ships to store the names of all the ships in a separate in-core lexicon. If not all allowable lexical entries are explicitly encoded, there will be terms encountered by the parser about which nothing is known. The problem is to assign these terms to a particular class, in the absence of a specific lexical entry.

Thus, given the sentence, *"Where is the Fox docked?"*, the parser would have to decide, in the absence of any prior information about "Fox", that it was the name of a ship, and not, say, a port.

III. Previous approaches

There are several methods by which unknown terms can be immediately assigned to a category: the parser can check the database to see if the unknown term is there (as in [Harris, 1977]); the user may be interactively queried (in the style of *RENDEVOUS* [Codd et al., 1978]); the parser might simply make an assumption based on the immediate context, and proceed (as in [Kaplan, 1979]). (We call these *extended-lexicon* methods.) However, these methods have the associated costs of time, inconvenience, and inaccuracy, and so constitute imperfect solutions.

Note in particular that simply using the database itself as a lexicon will not work in the general case. If the database is not fully indexed, the time required to search various fields to identify an unknown lexical item will tend to be prohibitive, if this requires multiple disk accesses. In addition, as noted in [Kaplan, Mays, and Joshi 1979], the query may reasonably contain unknown terms that are not in the database (*"Is John Smith an employee?"* should be answerable even if "John Smith" is not in the database).

IV. An Approach--Delay the Decision, then Compare Classification Methods

Our approach is to defer any lexical decision as long as possible, and then to apply the extended-lexicon methods identified above, in order of increasing cost.

Specifically, all possible parses are collected, using a semantic grammar (see below), by allowing the unknown term to satisfy any category required to complete the parse. The result is a list of categories for unknown terms, each of which is syntactically valid as a classification for the item. Consequently, interpretations that do not result in complete parses are eliminated. Since a semantic grammar tightly restricts the class of allowable sentences, this technique can substantially reduce the complexity of the remaining disambiguation process.

The category assignments leading to successful parses are then ordered by a procedure which estimates the cost of checking them. This ordering currently assumes an underlying cost model in which accessing the database on indexed or hashed fields is the least expensive, a single remaining interpretation warrants an assumption of correctness, and lastly, remaining ambiguities are resolved by asking the user.

A disambiguated lexical item is added temporarily to the in-core lexicon, so that future queries involving that term will not require repetition of the disambiguation process. After the item has not been referenced for some period of time (determined empirically) the term is dropped from the lexicon.

V. Example

This approach has been implemented in the parser for the Knowledge Base Management Systems (KBMS) project testbed. [Wiederhold, 1978] (The KBMS project is concerned with the application of artificial intelligence techniques to the design and use of database systems. Among other components, it contains a natural language front end for a CODASYL database in the merchant shipping domain.)

The KBMS parser is implemented using the *LIFER* package, a *semantic grammar* based system designed at SRI [Hendrix, 1977]. Semantic grammars have the property that the metasymbols correspond to objects and actions in the domain, rather than to abstract grammatical concepts. For example, the KBMS parser has classes called SHIPS and PORTS.

The KBMS parser starts with a moderate-size in-core lexicon (400 words); however, none of the larger database categories (SHIPS, PORTS, SHIPCLASSES, CARGOES) are stored in the in-core lexicon.

Following is a transcript from a run of the KBMS parser. The input to the parser is in italics; annotations are in braces.

←*is izmir in italy?* {"Italy" is known, from the in-core lexicon, to be a country. "Izmir" is unknown.}

```
> UNKNOWN TERM IZMIR
> POSSIBLE CATEGORIES: SHIPS, PORTS, CARGOES
```

{At the point where the word IZMIR is encountered, any category which admits a name is possible. These include ships, ports, and cargoes.}

```
> FINISHING PARSE
> POSSIBLE CATEGORY FOR IZMIR, LEADING TO VALID PARSE: SHIPS, PORTS
```

{When the parse is complete, the category "cargoes" has been eliminated, since it did not lead to a valid parse. So, the remaining two categories are considered.}

```
> CHECKING SHIPS FILE IN DATABASE
> IZMIR NOT THERE
> ASSUME THAT IZMIR IS A PORT.
```

{Of the two remaining categories, SHIPS is indexed in the database by name while PORTS is not and would therefore be very expensive to check. So, the SHIPS file is examined first. Since IZMIR is not in the database as a shipname, only PORTS remains. At this point, the parser assumes that IZMIR is a port since this is the only remaining plausible interpretation. This assumption will be presented to the user, and will ultimately be verified in the database query.}

```
> FINAL QUERY:
> For the PORTS with Portname equal to 'IZMIR',
> is the Portcountry equal to 'IT'?
```

A simple English generation system (written by Earl Sacerdoti), illustrated above, has been used to provide the user with a simplified natural language paraphrase of the query. Thus, invalid assumptions or interpretations made by the parser are easily detected. In a normal run, the information about lexical processing would not be printed.

In the example above, the unknown term happened to consist of a single word. In the general case, of course, it could be several words long (as is often the case with the names of ships or people).

Items recognized by extended-lexicon methods are added to the in-core lexicon, for a period of time. The time at which they are dropped from the in-core lexicon is determined by consideration of the time of last reference, and comparison of the (known) cost of recognizing the items again with the cost in space of keeping them in core.

VIII. Applications of this Method

The method of delaying a categorization decision until the parse is completed has some possible extensions. At the time a check is made of the database for classification purposes, it is known which query will be returned if the lookup is successful. For simple queries, therefore, it is possible not only to verify the classification of the unknown term, but also to fetch the answer to the query during the check of the database. For example, with the query *"What cargo is the Fox carrying?"*, the system could retrieve the answer at the same time that it verified that the "Fox" is a ship. Thus, the phases of parsing and query-processing can be combined. This 'pre-fetching' is possible only because the classification decision has been postponed until the parse is complete.

The technique of collecting all parses before attempting verification can also provide the user with information. Since all possible categories for the unknown term have been considered, the user will have a better idea, in the event that the parse eventually fails, whether an additional grammar rule is needed, an item is missing from the database, or a lexicon entry has been omitted.

VI. Limitations of this Method

In its simplest form, this method is restricted to operating with semantic grammars. Specifically, the files in the database must correspond to categories in the grammar. With a syntactic grammar, the method is still applicable, but more complicated; semantic compatibility checks are necessary at various points. Moreover, the set of acceptable sentences is not as tightly constrained as with a semantic grammar, so there is less information to be gained from the grammar itself.

This method (and all extended-lexicon methods) prevents use of an *INTERLISP*-type spelling corrector. Such a spelling corrector relies on having a complete in-core lexicon against which to compare words; the thrust of the extended-lexicon methods is the absence of such a lexicon.

If the unknown term already has a meaning to the system, which leads to a valid parse, the extended-lexicon methods won't even be invoked. For example, in the KBMS system, the question *"Where is the City of Istanbul?"* is interpreted as referring to the city, rather than the ship named 'City of Istanbul'. This difficulty is mitigated somewhat by the fact that semantic grammar restricts the number of possible interpretations, so that the number of genuinely ambiguous cases like this is comparatively small. For instance, the query *"What is the speed of the City of Istanbul"* would be parsed correctly as referring to a ship, since 'City of Istanbul' cannot meaningfully refer to the city in this case.

V. Conclusion

The technique discussed here could be implemented in practically any application that uses a semantic grammar—it does not require any particular parsing strategy or system. In the KBMS testbed, the work was done without any access to the internal mechanisms of LIFER. The only requirement was the ability to call user supplied functions at appropriate times during the parse, such as would be provided by any comparable parsing system.

This method was developed with the assumption that the costs of extended-lexicon operations such as database access, asking the user, etc., are significantly greater than the costs of parsing. Thus these operations were avoided where possible. Different cost models might result in different, more complex, strategies. Note also that the cost model, by using information in the database catalogue and database schema, can automatically reflect many aspects of the database implementation, thus providing a certain degree of domain-independence. Changes such as implementation of a new index will be picked up by the cost model, and thus be transparent to the design of the rest of the parser.

For natural language systems to provide practical access for database users, they must be capable of handling realistic databases. Such databases are often quite large, and may be subject to frequent update. Both of these characteristics render impractical the encoding and maintenance of a fixed, in-core lexicon. Existing systems have incorporated a variety of strategies for coping with these problems. This note has described a technique for reducing the number of lexical ambiguities for unknown terms by deferring lexical decisions as long as possible, and using a simple cost model to select an appropriate method for resolving remaining ambiguities.

VI. Acknowledgments

This work was performed under ARPA contract #N00039-80-G-0132. The Views and conclusions contained in this document are those of the authors and should not be interpreted as representative of the official policies, either expressed or implied, of DARPA or the U.S. Government. The authors would like to thank Daniel Sagalowicz, Norman Haas, Gary Hendrix and Earl Sacerdoti of SRI International for their invaluable assistance and for making their programs available to us. We would also like to thank Sheldon Finkelstein, Doug Appelt, and Jonathan King for proofreading the final draft.

VI. References

[1] Codd, E. F., et al., *Rendezvous Version 1: An Experimental English-Language Query Formulation System for Casual Users of Relational Data Bases*, IBM Research report RJ2144(29407), IBM Research Laboratory, San Jose, CA, 1978.

[2] Harris, L., *Natural Language Data Base Query: Using the database itself as the definition of world knowledge and as an extension of the dictionary*, Technical Report 77-2, Mathematics Dept., Dartmouth College, Hanover, NH, 1977

[3] Hendrix, G.G., *The LIFER Manual: A Guide to Building Practical Natural Language Interfaces*, Technical Note 138, Artificial Intelligence Center, SRI International, 1977

[4] Kaplan, S. J., *Cooperative Responses from a Portable Natural Language Data Base Query System*, Ph.D. dissertation, U. of Pennsylvania, available as HPP-79-19, Computer Science Department, Stanford University, Stanford, CA, 1979

[5] Kaplan, S. J., E. Mays, and A. K. Joshi, A Technique for Managing the Lexicon in a Natural Language Interface to a Changing Data Base, *Proc. Sixth International Joint Conference on Artificial Intelligence*, Tokyo, 1979, pp 463-465.

[6] Sacerdoti, E.D., Language Access to Distributed Data with Error Recovery, *Proc. Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, 1977, pp 196-202

[7] Waltz, D.L., An English Language Question Answering System for a Large Relational Database, *Communications of the ACM*, 21, 7, July, 1978

[8] Wiederhold, Gio, Management of Semantic Information for Databases, *Third USA-Japan Computer Conference Proceedings*, San Francisco, 1978, pp 192-197