COMMUNICATING WITH DATABASES IN NATURAL LANGUAGE

M. WALLACE, BA, MA, MSc, PhD

COMMUNICATING WITH DATABASES IN NATURAL LANGUAGE

M. WALLACE, BA, MA, MSc, PhD
Software Designer/Implementor
International Computers Limited
Reading, Berkshire, England

ELLIS HORWOOD LIMITED
Publishers · Chichester
Halsted Press: a division of
JOHN WILEY & SONS
New York · Chichester · Brisbane · Toronto

First published in 1984 by
ELLIS HORWOOD LIMITED
Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, England

The publisher's colophon is reproduced from James Gillison's drawing of the ancient Market Cross, Chichester.

Distributors:
Australia, New Zealand, South-east Asia:
Jacaranda-Wiley Ltd., Jacaranda Press,
JOHN WILEY & SONS INC.,
G.P.O. Box 859, Brisbane, Queensland 40001, Australia
Canada:
JOHN WILEY & SONS CANADA LIMITED
22 Worcester Road, Rexdale, Ontario, Canada.
Europe, Africa:
JOHN WILEY & SONS LIMITED
Baffins Lane, Chichester, West Sussex, England.
North and South America and the rest of the world:
Halsted Press: a division of
JOHN WILEY & SONS
605 Third Avenue, New York, N.Y. 10016, U.S.A.

British Library Cataloguing in Publication Data
Wallace, Mark
Communicating with databases in natural language. –
(Ellis Horwood series in artificial intelligence)
1. Prolog (Computer program language)
I. Title
001.64'24 QA76.73.P7

Library of Congress Card No.

ISBN 0-85312-639-9 (Ellis Horwood Limited)
ISBN 0-470-20105-3 (Halsted Press)

Typeset by Ellis Horwood Limited
Printed in Great Britain by R.J. Acford, Chichester

Table of Contents

Preface

This book is intended to give the reader an insight into the construction of a natural language front end.

The first chapter introduces natural language systems in general, but the remainder of the book concentrates on natural language (NL) front ends to databases. The second chapter attempts to bring out the essential issues by examining a variety of solutions implemented in existing NL front ends. The next four chapters focus in on a particular implementation, QPROC, in order to give the reader the more detailed understanding that cannot be gleaned from general surveys. In Chapter 4 it is presumed that the reader has some knowledge of logic, although all the logical constructs are fully illustrated with examples. In Chapter 5 QPROC's NL analysis is described but the linguistic concepts are introduced in non-technical terms. A section on data modelling assumes that the reader has some prior knowledge of the relational data model.

The last two chapters contain practical discussion – in Chapter 6 about the implementation of QPROC, in Chapter 7 about the current status and near term developments of NL front ends.

The whole book is spiced with examples of PROLOG programming to illustrate the implementation of QPROC and the suitability of PROLOG for the task. The book does not claim to be an introduction to PROLOG, but there is an appendix designed to bring the reader to a point where he can understand the example PROLOG programs, if such a text is not available to him.

That the book appeared at all can be traced back to Terry Smith and Roger Pikett of the BBC, and to Professor David Barron of Southampton University and Andrew Hutt of ICL, who agreed to the joint sponsorship of a Ph.D. on a

natural language database interface. I am very grateful to the SERC, ICL and the BBC for the funding of three years of research under David Barron's bolstering supervision; to Andrew Hutt who whirled me back to ICL where I have been pointed in the right direction, supported and encouraged by David Horth and Vince West; and to Ken Whitehead and Professor John Campbell who have read the book and who have both shown me a number of improvements which I have included in their entirety.

My initial interest in natural language understanding was sparked off and stoked by Alan Bond, at Queen Mary College, London. Throughout my research I have been continually encouraged and helped by my parents whose confidence in me will never be justified by anything I produce but was inevitably most inspiring!

Finally, to Julie who spent hours decoding my writing, typing and retyping it, I acknowledge a tremendous debt which cannot properly be expressed in this preface.

1

Introduction

1. NATURAL LANGUAGE PROCESSING

Computers never get jokes. In fact they can cope with so few of the functions of natural language that nobody has ever dared claim to have written a real natural language understanding system. What this book describes, and what perhaps one hundred groups of people are working on all over the world, are merely systems to deal with useful subsets of natural language. These groups are interested in natural language for a variety of different reasons. There is a purely research aspect and this is termed 'computational linguistics'. It is closely linked to computer science, linguistics, psychology and philosophy. Research has produced natural language (NL) systems to simulate a psychiatrist, to generate children's stories, to paraphrase and precis, and even to place a political interpretation on a set of facts.

The first practical application of NL was for machine translation from one language to another. At the moment this application is being heavily funded by the European Economic Community to cope with the seven official languages into which every document has to be translated! Currently there is still no machine translation, only machine-*aided* translation – the output of the computer is not a good translation and it needs to be edited by a linguist to make it idiomatic.

Machine translation systems have to cover quite a large vocabulary, without really 'understanding' the meaning of the subject matter. Naively, the system only has to know enough to find which word or phrase in the target language is closest to each word or phrase in the input.

Another area of application is for 'adviser' systems. Such systems need to be given a great deal of knowledge about a specific area so that they can detect the users' errors and explain them. These systems can appear extremely 'intelligent' as long as the conversation sticks closely to the application area. Surprisingly they have almost nothing in common with natural language translation systems, and an expert at programming machine translation systems might have very little idea how to set about building an adviser.

Natural language analysis is also required for automatic extraction of information from text. The information is then recorded in some standard form suitable to be input to a database, or efficiently searched by further software.

More divergent areas of development include natural language generation, so that computers don't have to talk gobbledygook, and processing spoken language, which is still at a rather speculative stage.

Major computer users are currently most interested, however, in natural language modules which fit on top of other computer software. Given the current interest in expert systems, a natural language 'front end' seems a very promising way to provide a man–machine interface to such complex software. In particular, natural language output to express the expert system's 'reasoning' would be very useful.

Many computer systems offer a range of tools for data extraction, statistical analysis and graphical output. Usually the tools have been developed quite independently and the user has to express similar commands to the different software packages in very different ways. A natural language 'front end' to all the different packages would enable the user to forget which package he† was using and how to address it. Thus, users are interested in natural language as a standard man–machine interface.

2. COMMUNICATING WITH DATABASES

The computer software best suited to benefit from a natural language front end is the database. Typically databases hold huge quantities of data which have to be stored in complex ways so as to secure the fastest access for the maximum proportion of queries. Many formal query languages have been designed to simplify the problem of getting the correct data out of this monolith.

These formal languages can be divided into 'one-dimensional' languages – composed of letters, numbers and mathematical signs – and 'two-dimensional' languages which enable the user to manipulate diagrams on a screen. A very good two-dimensional language is Query By Example [61]. An example query is

"List the products ordered by Good Co."

† I apologise for having used the third person pronoun in the masculine gender throughout!

(Words in lower case are typed on the screen by the user. The attribute names are generated by the system):

orderline	PRODUCTORDER	ORDER	QUANTITY	
	p. product	order		

order	ID	ORDERDATE	CUSTOMER
	order		cust

customer	ID	CUSTNAME	ADDRESS1	ADDRESS2	COUNTY	POSTCODE	. . .
	cust	good co					

(The database against which this query is being evaluated is the 'COPSE' database of Appendix 4).

"Despite the great practical differences between the many formal languages, linear or two-dimensional, many of the same criticisms apply to all", writes Cuff [7]. "In each system the user must form a query using not only a description of the data, but also a set of artificial syntactic, constructional and navigational elements to encase it."

Another problem is the requirement for completeness in a formal query language which unnecessarily complicates the definition of the language for casual users. Cuff writes, "Why emphasise a full syntax and precedence rules for combining logical operators, when they avoid complex queries and mishandle formal logic? . . . If some extra syntax is mandatory in order to eliminate ambiguity in certain infrequent cases, then it may be better to remove it . . . The ambiguity can be resolved by explaining the choices to the user when it arises".

Thirdly, a query formulation may be objectively much more complex than one would expect from the corresponding English question. For example in Query By Example, the query "List the products held at Bracknell" (against the COPSE database) is:

stock	PRODUCTSTOCK	STOCKWHSE	BINID	QTYONHAND	. . .
	p. product	bracknell			

while the previous example, "List the products ordered by Good Co." sounds no more complicated but requires three different tables.

Another pair of examples of queries expressed similarly in English but not (yet) in formal languages, is "What costs more than £50?" and "What costs more than desks?".

3. MAKING QUERY LANGUAGES MORE HABITABLE

Many of the shortcomings of formal query languages can be overcome by extending them and giving more intelligence to the interpreter.

By doing less with syntax and more with word meanings, the language can reduce the necessity for the user to conform to an artificial syntax. If the formal words have a meaning close to their natural meaning, the burden on the user is further reduced, especially if all alternative natural language words with the same meaning are allowable synonyms in the formal language. A final refinement would be to make the remaining syntax correspond to the syntax of natural language.

A second improvement would be to reduce the requirement for the user to know about the details of the database. Natural language synonyms for all database names are an obvious start, but also the facility for the interpreter to navigate around the database would be a great advantage (e.g. enabling the two Query By Example queries to be phrased in a similar way). The ability to use the structure of the data as well as the structure of the query to aid the interpretation of the user's input will be further discussed in the next chapter.

A third improvement that would be possible with an intelligent interpreter is to recognise the user's intended query on the basis of incomplete or slightly erroneous input. Thus simple queries could be recognised in a very abbreviated form (e.g. "Age Smith") although more complex queries such as "Which salesman had at least two customers ordering desks in June?" may need more complete expression. Queries with minor syntactic errors could be recognised if the remainder of the query was sufficiently unambiguous.

Another improvement would be to enable the user to phrase his queries incrementally. Thus his original query might be ambiguous and the system could prompt him to select which interpretation he intended. Or else the user could phrase a simple query ("How many customers are in debit?") and follow it up with supplementaries ("And who are they?").

Finally, an intelligent interpreter could perhaps recognise logical inconsistencies in a query and warn the user, or answer a broader but consistent query (e.g. "List orders placed by Good Co. and Better Co" could be interpreted as "List orders placed by either Good Co. or Better Co.").

Finally, two other desirable features of a query language are *curtness* and *breadth of use*.

Clearly all these are facilities which 'natural language front ends' are aiming to provide. In particular natural language is powerful and precise where required, but also curt, and it does provide a conversational interface which enables the user to ask incremental queries.

The description 'natural language' may be a drawback because it suggests capabilities beyond any current or foreseeable implementation. It tempts the user to ask common sense questions outside the area of the database's body of information, or evaluative questions within it. Tennant found that users tended to try a range of interactions outside the realm of database enquiry when provided with an NL interface [45].

The term 'natural language' front end is used for a good reason, however. It implies that the user should not consciously have to translate his queries into terms appropriate for the front end. Language users do unconsciously adapt their way of speaking to their audience, however, and the natural language subset accepted by the NL front end should be 'dense' enough for the user to click into it without conscious effort.

As long as a user does not have a vastly over-optimistic expectation of what the system can do for him, the intelligent natural language database front end can enable him to get the right answers with less frustration, and little requirement for technical support.

4. NL SYSTEM ARCHITECTURE

The modules of an NL database front end are

– the parser
– the formal query generator
– the database access routines.

It is not sensible, however, for the system first to perform a parse, then to construct a query and then to access the data. The grammatical analysis of a sentence is closely dependent upon its meaning, so the grammar must be tailored to the system's knowledge structure. With the development of data modelling and the implementation of data dictionaries it has become possible automatically to tailor the parser to a particular application by writing into the parser calls on the data dictionary.

Thus the parser uses the same information as the query generator and so formal queries can be built up during the parse. In fact, if a formal query cannot be built up from the input sentence, the parse fails. This places a burden on the formal query language. It must be powerful enough to express even queries which, for implementational reasons, cannot be executed against the data. A further requirement is for the formal queries to yield sensible answers. Thus "Is the Tory candidate at Worthing over 50?" should get a more sensible answer than "no" if there is no Tory candidate at Worthing. In fact the QPROC system described in this book actually evaluates the formal interpretation of definite noun phrases (like "the Tory candidate at Worthing") against the database before completing the parse of the input query.

It follows that the formal language in the QPROC system is the fulcrum on which the parser and database access routines are pivoted. The discussion of the parser is to a considerable degree expressed in terms of the formal subqueries generated by the different grammatical constructions.

Another fundamental feature of the design of an NL database front end is application independence. There is a trade-off between the size of the NL subset the system covers, and its adaptability to new databases. An NL interface *builder* is a system whose grammar and formal query generator must be completely rewritten for each new application. Such a system provides useful tools for constructing these components. The tools used have a major impact on the character of the generated system. LIFER [22] is an NL interface builder, and the systems it generates are very impressive.

Other systems provide considerable adaptability without any rewriting of the parser. A very good example is TEAM [19] which includes an 'acquisition component', enabling a database expert to tailor the system to a new application by simply answering questions about the structure and meaning of items in the database.

The QPROC system is more similar to TEAM, though considerably less comprehensive.

5. THE SCOPE OF THE IMPLEMENTATION

J. L. Austin roughly divided sentences into five categories:

(1) Objective judgements
(2) Ordering, advising, questioning
(3) Promising, understanding
(4) Social behaviour – apologising, congratulating
(5) Taking a stance – replying, assuming, conceding.

Of all these, an NL database front end can only deal with a small proportion of judgements, orders and questions. There is considerable research on the analysis of speech acts which throws further light on (2) and (5), but this will not be dealt with in the discussion of the current implementation.

Another research topic may be termed 'knowledge representation'. Firstly this encompasses the structure required to express facts, opinions, etc., and secondly it must include work on the objects of knowledge. Thus, for example, if disparate statements are to be interpreted as giving knowledge about the same thing, then the research must investigate the common primitive concepts underlying the different phrasings. In this book the 'knowledge representation' is simply the database and data 'model', and only current database concepts are used.

NL front ends, then, have the limited objective of coping with as 'habitable' as possible a natural language subset, given the available data modelling concepts. The design is, in a sense, bottom-up, starting with the data model, superimposing as powerful as possible a query language, and then using linguistic tools to map natural language sentences onto formal language.