Technical Report

UCAM-CL-TR-24
ISSN 1476-2986

Number 24

Steps towards natural language to data language translation using general semantic information

B.K. Boguraev, K. Spärck Jones

March 1982

ISSN 1476-2986

Steps towards natural language to data language translation using general semantic information

B.K. Boguraev and K. Sparck Jones
Computer Laboratory, University of Cambridge
Corn Exchange Street, Cambridge CB2 3QG, England

March 1982

Paper accepted for ECAI-82, the 1982 European Conference on Artificial Intelligence, Orsay, France.

Abstract

The aim of the work reported here is to maximise the use of general semantic information in an AI task processor, specifically in a system front end for converting natural language questions into formal database queries. The paper describes the translation component of such a front end, which is designed to work from the question meaning representation produced by a language analyser exploiting only general semantics and syntax, to a formal query relying on database-specific semantics and syntax. Translation is effected in three steps, and the paper suggests that the rich and explicit meaning representations using semantic primitives produced for input sentences by the analyser constitute a natural and effective base for further processing.

Keywords - natural language, databases, semantics, primitives

1. Introduction

Relatively efficient front ends for natural language access to databases can be provided [1,2]. But these front ends are effective because their language analysers are biased towards the database universe of discourse. The lexicon is specialised, and a database-specific semantic grammar may be used.

Such front ends can be stigmatised because they lack portability from one database to another, and because they fail to take advantage of the powerful general semantic apparatus which must underlie much ordinary language use. However they recognise the fact that the world of any database is a specialised world, and that questions addressed to it may be correspondingly idiosyncratic. Any attempt to make a language analyser for database questions more general has therefore to be matched by the provision of means for linking general semantic characterisations of input words and sentences with database-specific characterisations of data language terms and expressions.

Our current work on this problem is described here. A database system front end is being developed combining a general semantic analyser for interpreting natural language input questions with a specific translator for deriving formal language search queries. The claim made is that the nature of the question meaning representation produced by the analyser, though designed for general purposes without reference to the database access task, is nevertheless well-suited to the database application. The form and content of the meaning representation are explicit and rich enough to support natural and simple query derivation operations.

The paper describes the nature of the translator, and illustrates the output of the successive translation operations so far implemented. Eventually, to interface with an existing database management system, a formal query may have to be converted into a locally appropriate low-level search specification referring to the storage organisation of the data. This is not discussed here. The paper focusses on the transition from natural language to high-level query language.

2. The translator design

The translation process applied to the question meaning representation produced by the analyser has three aspects, for convenience in program development treated as three successive processing steps: extraction of 'elementary propositions'; construction of quantified expressions; and substitution of database concept names. The first of these is not database-specific, but may be task-specific; the second may be influenced, as in the LUNAR project [3], by the database; the third is wholly database-dependent. All of these steps rely on the properties of the question meaning representation, i.e. work with the analyser's meaning representation language. The presupposition underlying the translator design is that the same, or a closely-related meaning representation language can be used to characterise the terms and expressions of the data query language.

3. The question meaning representation

The system language analyser has been fully described elsewhere [4,5]. It is sufficient here to note that both conventional syntactic information and a range of semantic pattern types are applied by an ATN processor to identify word senses and sentence structures.

The nature of the output meaning representation can only be summarised and illustrated here [6]. Its most relevant properties are that word senses are characterised by formulae using semantic category primitives, while sentence structures are characterised by dependency trees using semantic relation primitives, that is, case labels. The category primitives are like Wilks' [7], but sentence structure characterisations are more explicit and more complex than his: the dependency trees are more constrained, systematically labelled, properly hierarchical, and also succinct, structures than Wilks'.

Thus the essential feature of the sentence structure characterisation is that while sentence elements are directly and simply treated as case role holders, case structures are also directly and simply related to one another by case links or by common elements. The representation as a whole is straightforwardly expressive, and is therefore easy to search and manipulate, particularly since elements common to different case substructures are concisely but conveniently connected by trace pointers. The illustration of Figure 1a shows an example question representation organised round its highest-level verb with its case roles (marked by @@), with the latter in turn filled either by formulae for word senses, or by trace pointers to case role fillers of dependent substructures organised in a similar way.

4. Query derivation

In general in database access, the properties of the database to be searched make it natural to treat the formal query as a more-or-less conventional predicate logic expression constructed from simple, or atomic, propositions. This approach was early followed in LUNAR, and is currently exemplified by Warren and Pereira [8]. The analyser meaning representations described above are well-suited to the extraction of such query building blocks. The essentials of this first translation step are as follows. Each verb and its case role fillers can be regarded as generating one or more 'SVO' expressions. These fall into two classes, according to the broad characteristics of the verb and its case fillers, as defined by the case labels and by the head primitives in the formulae for the word senses filling the slots. Thus verbs can be characterised as either of LINK or POSS type, and slot fillers as of OBJ or PROP type (these terms are deliberately distinguished from "entity", "attribute". etc., which categorise data world concepts). The SVO-type elementary propositions, called triples, are then either of [OBJ LINK OBJ] or [OBJ POSS PROP] type.

Further, the ways in which triples can be dependent on one another in the sentence representation are quite limited: the 'subject' or 'object' of a dependent triple can be either 'subject' or 'object' of the governing triple. (Another type of connection holds between several triples at the same level of verb dependency.) It is therefore easy to extract the equivalent set of connected triples from the given dependency structure, and a simple set of rules has so far been found adequate for the purpose. The first, triple extraction, phase of translation processing thus consists essentially of a reorganisation and marking of the initial structure: the structure's elements are defined as OBJ, LINK, etc., and the detailed formula and case information associated with them is carried along for future utilisation. The triples for the example sentence, in a summarily abbreviated form retaining only word-sense names, are shown in Figure 1b.

The second stage is to generate a quantified structure for the formal query. Currently, both the individual quantifiers onto which natural language words are mapped, and the overall form of quantified expressions, follow the LUNAR model. Again, given the limited ways in which two triples can be connected, only a few rules are required to derive appropriate LUNAR-style structures with quantifier, variable of quantification, its class range, any class restrictions, and quantified proposition or command, from the complete triple information. (In this, determiner information directly attached to relevant elements of the meaning representation, and carried forward in the triples, is exploited.) The results for the example sentence are illustrated in Figure 1c, again with the information associated with individual word senses abbreviated.

In both these steps, testing so far suggests that the meaning representation makes the subsequent processing much cleaner than would be the case with a representation dominated by conventional syntax, as in LUNAR and in Warren and Pereira's system.

The third step in query construction is to replace the natural language inputs represented by the terms of the atomic propositions with the corresponding data language ones. The natural language words are represented by their meaning language formulae. The hypothesis is that critical data language terms, like entity type names, attribute names, and relation names, though not necessarily value names, are similarly represented in either the same meaning representation language or one closely related to it. Thus though data language words may not have exactly the same meaning as their related natural language words, their common character can be captured and exploited in the final translation step. The internal structures of the atomic propositions derived from natural language can be similarly mapped onto ones appropriate to the database using word types and case labels, via a repertoire of proposition forms.

Thus if the first translation step presupposes only the re-formation of the initial meaning representation appropriate to a certain type of task, the quantified expression structures generated by the second step are already in a database-oriented meaning representation language, but probably reflect common features of many database meaning representation languages. It is in the third step that the detailed connection is made with the individual database. This constitutes the hardest step in the process, but the project assumes (since this step had not yet been implemented), that the substitution of data language for natural language expressions here will be facilitated by the power of the meaning representation tools available, and by the fact that the substitution is focussed on the elementary propositions within the query structure built by the second stage. The philosophy underlying the translator broadly resembles that of PHLIQA1 [9], but the substantive details are different, chiefly because an explicit extra transition has to be made from general semantic meaning representations to domain-specific ones.

5. Conclusion

Database access, though challenging, is only one language-using task. The meaning representations provided by our analyser are designed, as those originally proposed by Wilks were designed, to capture general semantic properties of text, giving rich and explicit characterisations of text meaning in principle adequate for a variety of tasks. The representations should therefore be evaluated in use for different purposes, preferably in competition with Schank's Conceptual Dependency representations, which are the most obvious alternatives. However, the database access project is a serious test of the effectiveness of the analyser's meaning representations, since many of the requirements to be met in this case would also have to be met in the context of other tasks.

References

1. Harris, L.R. 'Experience with ROBOT in 12 commercial natural language data base query application', IJCAI-79, 1979, 365-368.

2. Damerau, F.J. 'The transformation question answering (TQA) system:

description, operating experience, and implications', Report RC8287, IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y., 1980.

3. Woods, W.A. 'Semantics and quantification in natural language question answering', Advances in Computers, 17, 1978, 1-87.

4. Boguraev, B.K. Automatic resolution of linguistic ambiguities, Technical Report No. 11, Computer Laboratory, University of Cambridge, 1979.

5. Boguraev, B.K. and Sparck Jones, K. 'A natural language analyser for database access', Information Technology: Research and Development, 1, 1982, 23-39.

6. more details appear in 4, and in K. Sparck Jones, 'Basic semantics information', an internal memo.

7. Wilks, Y.A. 'Good and bad arguments about semantic primitives', Communication and Cognition, 10, 1977, 53-74.

8. Warren, D.H.D. and Pereira, F.C.N. 'An efficient easily adaptable system for interpreting natural language queries', RP 155, Department of Artificial Intelligence, University of Edinburgh, 1981.

9. Bronnenberg, W.J.H.J. et al. 'The question answering system PHLIQA1', in Natural language question answering systems (Ed. Bolc), London: Macmillan, 1979.

a) question meaning representation

```
Sentence:  WHAT IS THE WEIGHT OF ALL RED PARTS WHICH ARE SUPPLIED BY CLARK
              TO ROME?

(clause
   (type question)
   (tns present)
   (v
      (be2
         ((*ent subj) (((own state) obje) be))
         (@@ agent
            ((trace (clause v obj))
               (clause
                  (type relative)
                  (tns present)
                  (aspect (passive))
                  (v
                     (supply1
                        ((*org subj) ((*inan obje) ((*org recipient)
                                 (((((@recipient subj) have) cause) goal)
                                    give))))
                        (@@ agent (Clark (mal (indiv man))))
                        (@@ obj
                           ((trace (clause v agent))
                              (clause
                                 (v
                                    (be2
                                       ((*ent subj) (((own state) obje) be))
                                       (@@ agent
                                          (part1
                                             ((*inan poss)
                                                ((work goal) (subj thing)))
                                             (@@ number many)
                                             (@@ det (all1 (all)))) )
                                       (@@ state
                                          (colour)
                                          (val
                                             (red1
                                                ((*inan poss)
                                                   (((man subj) (see sense))
                                                         (obje kind)))) ))) ))) )
                        (@@ location (Rome (this (where point)))) ))) ))
         (@@ state
            (weight ((*ent poss) (count sign)))
            (val (query (dummy)))) )))
```

b) question triples

```
                  & [$Obj2(part1) $Poss2(be2) $Prop2(weight=query)]
               & [$Obj2(part1) $Poss1(be2) $Prop1(colour=red1)]
            & [$Obj1(Clark) $Link1(supply1) $Obj2(part1)]
            & [$Obj1(Clark) $Link1(supply1) $Obj3(Rome)]
```

c) question quantified expression

```
(For Every $Var1/part1
   : (AND
        (colour $Var1 red1)
        (For The $Var2/Clark
           - (supply1 $Var2 $Var1))
        (For The $Var3/Rome
           - (supply1 $Var1 $Var3)))
   - (Display (weight $Var1)  ))
```

Figure 1. Analyser and translator outputs