BusTUC - A natural language bus route adviser in Prolog

Tore Amble
Knowledge Systems Group
Department of Computer and Information Science
NTNU
amble@idi.ntnu.no

25. april 2000

Sammendrag

The paper describes a natural language based expert system route adviser for the public bus transport in Trondheim, Norway. The system is available on the Internet, and has been installed at the bus company's web server since the beginning of 1999. The system is bilingual, relying on an internal language independent logic representation.

1 Introduction

A natural language interface to a computer database provides users with the capability of obtaining information stored in the database by querying the system in a natural language (NL). With natural language as a means of communication with a computer system, the users can make a question or a statement in the way they normally think about the information being discussed, freeing them from having to know how the computer stores or processes the information.

The present implementation represents a a major effort in bringing natural language processing into practical use. A system is developed that can answer queries about bus routes, stated in natural language texts, and made public through the Internet World Wide Web (`www.idi.ntnu.no/bustuc/`).

Trondheim is a small city with a university and 140000 inhabitants. The central bus system in Trondheim has 42 bus lines, serving 590 stations, with

1900 departures per day (in average). That gives approximately 60000 scheduled bus station passings per day, which is somehow represented in the route data base.

The starting point is to automate the function of a route information agent. The following example of a system response is taken from an actual request over telephone to the local route information company:

```
Hi, I live in Nidarvoll and tonight I must
reach a train to Oslo at 6 oclock.
```

A typical answer would follow quickly:

```
-----------------------------------------------
Bus number 54 passes by Nidarvoll school at 1710
and arrives at Trondheim Railway Station at 1725.
-----------------------------------------------
```

In between the question and the answer is a process of lexical analysis, syntax analysis, semantic analysis, pragmatic reasoning and database query processing and answer generation.

One could argue that the information content could be solved by an interrogation, whereby the customer is asked to produce 4 items: `departure station, arrival station, earliest and latest arrival time`. It is a myth that natural language is better way of communication because it is "natural language". The challenge is to prove by demonstration that an NL system can be made that will be preferred to the interrogative mode. To do that, the system has to be correct, user friendly and almost complete within the actual domain.

2 Previous Efforts, CHAT-80, PRAT-89 and HSQL

The system, called BusTUC is built upon the classical system CHAT-80 ([WP82]). CHAT-80 was a state of the art natural language system that was impressive on its own merits, but also established Prolog as a viable and competitive language for Artificial Intelligence in general. The system was a brilliant masterpiece of software, efficient and sophisticated. The natural language system was connected to a small query system for international geography. The following query could be analysed and answered in less than half a second:

```
Which country bordering the Mediterranean borders a
country that is bordered by a country whose population
exceeds the population of India?
```

(*The answer 'Turkey' has become incorrect as time has passed. The irony is that Geography was chosen as a domain without time.*)

The ability to answer ridiculously long queries is of course not the main goal. The main lesson is that complex sentences are analysed with a proper understanding without sacrificing efficiency. Any superficial pattern matching technique would prove futile sooner or later.

2.1 Making a Norwegian CHAT-80, PRAT-89

At the University of Trondheim (NTNU), two students made a Norwegian version of CHAT-80,called PRAT-89 ([TV88],[TV89]). (Also, a similar Swedish project SNACK-85 was reported).

The dictionary was changed from English to Norwegian together with new rules for morphological analysis. The change of grammar from English to Norwegian proved to be amazingly easy. It showed that the langauges were more similar than one would believe, given that the languages are incomprehensible to each other's communities.

After changing the dictionary and grammar, the following Norwegian query about the same domain could be answered correctly in a few seconds.

```
Hvilke afrikanske land som har en befolkning større
enn 3 millioner og mindre enn 50 millioner og er nord
for Botswana og øst for Libya har en hovedstad som
har en befolkning større enn 100 tusen ?
```

("Which African countries that have a population greater than 3 millions and less than 50 millions and is north of Botswana and east of Libya has a capital which has a population greater than 100 thousands ?")

2.2 HSQL - Help System for SQL

A Nordic project HSQL (Help System for SQL) was accomplished in 1988-89 to make a joint Nordic effort interfaces to databases.

The HSQL project was led by the Swedish State Bureau (Statskontoret), with participants from Sweden, Denmark, Finland and Norway [AKL$^+$90]. The aim of HSQL was to build a natural language interface to SQL databases for the Scandinavian languages Swedish, Danish and Norwegian. These languages are very similar, and the Norwegian version of CHAT-80 was easily extended to the other Scandinavian languages. Instead of Geography, a more typical application area was chosen to be a query system for hospital administration. We decided to target an SQL database of a hospital administration which had been developed already.

The next step was then to change the domain of discourse from Geography to hospital administration, using the same knowledge representation techniques used in CHAT-80. A semantic model of this domain was made, and then implemented in the CHAT-80 framework.

The modelling technique that proved adequate was to use an extended Entity Relationship (ER) model with a class (type) hierarchy, attributes belonging to each class, single inheritance of attributes and relationships.

Connecting the system to an SQL database.

After the remodelling, the system could answer queries in "Scandinavian" to an internal hospital database as well as CHAT-80 could answer Geography questions. HSQL produced a Prolog-like code FOL (First Order Logic) for execution. A mapping from FOL to the data base Schema was defined, and a translator from FOL to SQL was implemented. The example

```
Hvilke menn ligger i en kvinnes seng?
```

(*"Which men lie in a woman's bed?"*)

was translated dryly into the SQL query:

```
SELECT DISTINCT T3.name,T1.sex,T2.reg_no,T3.sex,
       T4.reg_no,T4.bed_no,T5.hosp_no,T5.ward_no
FROM PATIENT T1,OCCUPANCY T2,PATIENT T3,
     OCCUPANCY T4,WARD T5
WHERE (T1.sex='f') AND (T2.reg_no=T1.reg_no) AND
      (T3.sex='m') AND (T4.reg_no=T3.reg_no) AND
      (T4.bed_no=T2.bed_no) AND (T5.hosp_no=T4.hosp_no) AND
      (T5.ward_no=T4.ward_no)
```

2.3 The Understanding Computer

The HSQL was a valuable experience in the effort to make transportable natural language interfaces. However, the underlying system CHAT-80 restricted the further development.

After the HSQL Project was finished, an internal reseach project TUC (The Understanding Computer) was initiated at NTNU to carry on the results from HSQL. The project goals differed from those of HSQL in a number of ways, and would not be concerned with multimedia interfaces. On thc other hand, portability and versatility were made central issues concerning the generality of the language and its applications. The research goals could be summarised as to

- Give computers an operational understanding of natural language.
- Build intelligent systems with natural language capabilities.
- Study common sense reasoning in natural language.

A test criterion for the understanding capacity is that after a set of definitions in a Naturally Readable Logic, NRL, the system's answer to queries in NRL should conform to the answers of an idealised rational agent.

```
Every man that lives loves Mary. John is a man. John lives.
Who loves Mary?
==> John
```

NRL is defined in a closed context. Thus interfaces to other systems are in principle defined through simulating the environment as a dialogue partner.

TUC is a prototypical natural language processor for English written in Prolog. It is designed to be a general purpose easily adaptable natural language processor. It consists of a general grammar for a subset of English, a semantic knowledge base, and modules for interfaces to other interfaces like UNIX, SQL-databases and route information services.

2.4 The TABOR Project

It so happened that a Universtity project was starteded in 1996, called TABOR ("Speech based user interfaces and reasoning systems"), with the aim of building an automatic public transport route oracle, available over the public telephone. At the onset of the project, the World Wide Web was fresh, and not as widespread as today, and the telephone was still regarded as the main source of information for the public. Since then, the Internet has become the dominant medium, and it is as likeley to find a computer with Internet connection, as finding a telephone, or a local busroute booklet for that matter.

It was decided that a text based information system should be built, regardless of the status of the speech rocgnition and speech synthesis effort, which proved to lag behind after a while.

The BusTUC system

The resulting system BusTUC grew out as a natural application of TUC, and an English prototype could be built within a few months ([Bra97]). Since the summer 1996, the prototype was put onto the Internet, and has been developed and tested more or less continually since then. The most important extension was that the system was made bilingual (Norwegian and English) during the fall 1996.

In the spring 1999, the BusTUC was finally adopted by the local bus company in Trondheim (A/S Trondheim Trafikkselskap), which set up a server (300 MHz PC with Linux).

Until today, over 150.000 questions have been answered, and BusTUC seems to stabilize and grow increasingly popular.

3 Anatomy of the bus route oracle

The main components of the bus route information systems are:

- A parser system, consisting of a dictionary, a lexical processor, a grammar and a parser.
- A knowledge base (KB), divided into a semantic KB and an application KB
- A query processor, containg a routing logic system, and a route data base.

The system is bilingual and contains a double set of dictionary, morphology and grammar. Actually, it detects which language is most probable by counting the number of unknown words related to each language, and acts accordingly. The grammars are surprisingly similar, but no effort is made to coalesce them. The Norwegian grammar is slightly bigger than the English grammar, mostly because it is more elaborated but also because Norwegian allows a freer word order.

3.1 Features of BussTUC

For the Norwegian system, the figures give an indication of the size of the domain: 420 nouns, 150 verbs, 165 adjectives, 60 prepositions, etc.

There are 1300 grammar rules (810 for English) although half of the rules are at a low lexical level.

The semantic net described below contains about 4000 entries.

A big name table of 3050 names in addition to the official station names, is required to capture the variety of nameings. A simple spell correction is part of the system (essentially 1 character errors).

The pragmatic reasoning is needed to translate the output from the parser to a route database query language. This is done by a production system called Pragma, which acts like an advanced rewriting system with 580 rules.

In addition, there is another rule base for actually generating the natural language answers (120 rules).

The system is mainly written in Prolog (Sicstus Prolog 3.7), with some Perl programs for the communication and CGI-scripts.

At the moment, there are about 35000 lines of programmed Prolog code (in addition to route tables which are also in Prolog). Sicstus Prolog proved to be extremely efficient and reliable for the application.

Average response time is usually less than 2 seconds, but there are queries that demand up to 10 seconds.

The error rate for the single, correct, complete and relevant questions is about 2 percent.

3.2 The Parser System

The Grammar System

The grammar is based on a simple grammar for statements, while questions and commands are derived by the use of movements. The grammar formalism which is called Consensical Grammar, (CONtext SENSItive CompositionAL Grammar) is an easy to use variant of Extraposition Grammar ([PW80]), which is a generalisation of Definite Clause Grammars. Semantically, a phrase is composed of the semantics of the subphrases; the basic constituents being generalized verb complements. As for Extraposition grammars, a grammar is translated to Definite Clause Grammars, and executed as such.

A characteristic syntactic expression in Consensical Grammar may define an incomplete construct in terms of a "difference " between complete constructs. This implements various kinds of movements by using the subtracted parts instead of reading from the input, immediately or after a gap.

The effect is the same as for Extraposition grammars, but this format allows a more intuitive reading. Examples of grammar rules:

```
statement(P) --->
   noun_phrase(X,VP,P),
   verb_phrase(X,VP).

statement(Q) --->
```

```
      verb_complements0(VC),  % initial optional verb_complements
      statement(Q) -...       % may be  inserted  after a gap
           verb_complements0(VC).

   whoseq(P) --->    % whose dog barked?
      [whose],
      noun(N),
      whoq(P) - ([who],[has],[a],noun(N),[that]). % without gap

   whoq(P) --->
      [who],
      whichq(P) - ([which],[person]).

   whichq(which(X)::P) --->
      [which],
      statement(P) - the(X).
```

Example:

```
      Whose dog barked?
```

is analysed as if the sentence had been

```
    Who has a dog that barked?
```

which is analysed as

```
    Which person has a dog that barked?
```

which is analysed as

```
    for which X is it true that
    the (X) person has a dog that barked?
```

where the last line is analysed as a `statement`.

Movement is easily handled in Consensical Grammar without making special phrase rules for each kind of movement. The following example shows how TUC manages a variety of analyses using movements:

```
    Max said Bill thought Joe believed Fido Barked.

    Who said Bill thought Joe believed Fido barked?    ==> Max
    Who did Max say thought Joe believed Fido barked?  ==> Bill
    Who did Max say Bill thought believed Fido barked? ==> Joe
```

The parser

The experiences with Consensical grammars are a bit mixed however. The main problem is the parsing method itself, which is top down with backtracking. Many principles that would prove elegant for small domains turned out to be to costly for larger domains, due to the wide variety of modes of expressions, the incredible ambiguity and the sheer size of the covered language.

These problems also made it imperative to introduce a timeout on the parsing process of embarassing 10 seconds. Although most sentences would be parsed within a second, some legal sentences of moderate size actually need this time.

The disambiguation is a major problem for small grammars and large languages, and was solved by the following guidelines:

- a semantic type checking was integrated into the parser, and would help to discard semantically wrong parses from the start.
- a heuristics proved almost irreproachable: The longest possible phrase of a category that is semantically correct is in most cases the preferred interpretation.
- due to the perplexity of the language, some committed choices (cuts) had to be inserted into the grammar at strategic places. As one could fear however, this implied that wrong choices being made at some point in the parsing could not be recovered by backtracking.

3.3 The semantic knowledge base

Adaptability means that the system does not need to be reprogrammed for each new application.

The design principle of TUC is that most of the changes are made in a tabular semantic knowledge base, while there is one general grammar and dictionary. In general, the logic is generated automatically from the semantic knowledge base.

The nouns play a key role in the understanding part as they constitute the class or type hierarchy. Nouns are defined in an `a-kind-of` hierarchy. The hierarchy is tree-structured with single inheritance. The top level also constitute the top level ontology of TUC's world.

In fact, a type check of the compliances of verbs, nouns adjectives and prepositions is not only necessary for the semantic processing but is essential for the disambiguation in the syntax analysis. In TUC, a declaration of the legal combinations are carefully assembled in the semantic network, which then serves a dual purpose. These semantic definitions are necessary for disambiguating prepositional attachments, for instance in the following sentences

```
The dog saw a man with a telescope.
The man saw a dog with a telescope.
```

to be treated differently because `with telescope` may modify the noun `man` but not the noun `dog`, while `with telescope` modifies the verb `see`, restricted to person.

3.4 The Query Processor

Event Calculus

The semantics of the phrases are built up by a kind of verb compelements, where the event play a central role.

The text is translated from Natural language into a form called TQL (Temporal Query Language/TUC Query Language) which is a first order event calculus expression, a self contained expression containing the literal meaning of an utterance.

The formalism TQL that was defined, inspired by the Event Calculus by Kowalski and Sergot ([KS86]). The TQL expressions consist of predicates, functions, constants and variables. The textual words of nouns and verbs are translated to generic predicates using the selected interpretation. The following question

```
Do you know whether the bus goes to Nidarvoll on Saturday ?
```

would give the TQL expression below. Typically, the Norwegian equivalent

```
Vet du om bussen går til Nidarvoll på søndag ?
```

gives exactly the same code.

```
test::                            % Type of question
  isa(real,program,bustuc),       % bustuc is a real program
  isa(real,bus,A),                % A is a real bus
  isa(real,saturday,B),           % B isa saturday
  isa(real,place,nidarvoll),      % nidarvoll isa place
  event(real,D),                  % D is an event
  know(id,whether,bustuc,C,D),    % C is a statement known at D
  event(C,E),                     % E is an event in C
```

```
action(go,E),                   % the action of E is 'go'
actor(A,E),                     % the actor of E is  A
srel(to,place,nidarvoll,E),     % E is related to nidarvoll
srel(on,time,B,E).              % E is related on the saturday B
```

The event parameter plays an important role in the semantics. It is used for various purposes. The most salient role is to identify a subset of time and space in which an action or event occured. Both the actual time and space coordinates are connected to the actions through the event parameter.

Pragmatic reasoning

The TQL is translated to a route database query language (BusLOG) which is actually a Prolog program. This is done by a production system called Pragma, which acts like an advanced rewriting system with 580 rules.

4 Conclusions

The TUC approach has as its goal to automate the creation of new natural language interfaces for a well defined subset of the language and with a minimum of explicit programming.

The implemented system has proved its worth, and is interesting if for no other reason. There is also an increasing interest from other bus companies and route information companies alike to get a similar system for their customers.

Further work remains to make the parser really efficient, and much work remains to make the language coverage complete within reasonable limits.

It is an open question whether the system of this kind will be a preferred way of offering information to the public.

If it is, it is a fair amount of work to make it a portable system that can be implemented elsewhere, also connecting various travelling agencies.

If not, it will remain a curiosity. But anyway, a system like this will be a contribution to the development of intelligent systems.

Referanser

[AKL$^+$90] Tore Amble, Erik Knudsen, Aarno Lehtola, Jan Ljungberg, and Ole Ravnholt. *Naturlig Språk och Grafik - nya vägar inn i databaser.* Statskontoret, 1990. Rapport om HSQL, ett kunskapsbaseret hjälpsystem för SQL.

[Bra97] Jon S. Bratseth. BusTUC - A Natural Language Bus Traffic Informations System. Master's thesis, The Norwegian University of Science and Technology, 1997.

[KS86] R. Kowalski and M. Sergot. A logic based calculus of events. *New Generation Computing*, 8(0):67–95, 1986.

[PW80] F.C.N. Pereira and D.H.D. Warren. Definite clause grammar for language analysis. *Artificial Intelligence*, 0(3), 1980.

[TV88] J. Teigen and V. Vetland. Syntax analysis of norwegian language. Technical report, The Norwegian Institute of Technology, 1988.

[TV89] J. Teigen and V. Vetland. Handling reasonable questions beyond the linguistic and conceptual coverage of
natural language interfaces. Master's thesis, The Norwegian Institute of Technology, 1989.

[WP82] D.H.D Warren and F.C.N. Pereira. An efficient and easily adaptable system for interpreting natural language queries. *Computational Linguistics*, 8(3-4), 1982.