Parsing in OntoQuery – Experiments with LKB

Patrizia Paggio
Center for Sprogteknologi (CST)
patrizia@cst.ku.dk

1 Introduction

The idea that Natural Language Processing (NLP) may be helpful to Information Retrieval (IR) is not a new one. Already in the late '60s and early '70s, experiments were made to automatically identify references to thesaurus concepts and complex terms in documents (reported in (Allan 2000)). Experiments with the use of NLP e.g. for query expansion and complex phrase detection continued – albeit with scarce success – throughout the '80s. In 1996, a special track was dedicated to NLP at the recurring Text REtrieval Conference (TREC-5) (see (Strzalkowski, Lin & Perez-Carballo 1998)). The four participating sites claimed up to 40% accuracy improvements due to the use of NLP techniques. Especially query expansion and nominal phrase (NP) identification seemed to have provided the most notable gains compared to term-only runs. Although it can be argued, as Allan does, that the same accuracy can be achieved by way of statistical means, and that NLP is still too error-prone to work well for large texts, TREC-5 certainly contributed to a growing interest in the use of NLP in IR. In fact, basic NLP techniques such as tokenising, stemming, and the use of stopword lists are fairly common in IR systems. Some systems also use more advanced but still relatively robust NLP, such as phrase identification and named entity extraction. Recently, NLP has again come into focus in connection with the question answering track at TREC-8 (1999), as systems using NLP techniques actually outperformed statistics-based ones. In particular, (Voorhees & Tice 2000) report that the best scoring system for the 50 byte limit task[1] was the *Textract* system

[1] In the QA track, systems had to match user queries with relevant passages no longer than either 50 or 250 bytes.

developed at Cymfony Inc. (Srihari & Li 2000), which uses named entity tagging and shallow parsing to extract templates from queries and texts.

In the OntoQuery project, matching of the user queries with the relevant retrieved texts (or text fragments) is guided by the semantic analysis of nominal phrases (NP's). Therefore, one of the lines of research pursued in the project concerns the possible use of NLP to identify NP's and assign them an adequate interpretation. To this end, several techniques will certainly be useful. In the first project prototype, we are experimenting with a pipeline architecture where queries and text to be indexed go through the following NLP analysis steps:

- Part-of-speech tagging, which assigns each word a unique syntactic category. The POS tagger used is Brill's tagger (Brill 1995), which has been trained to tag Danish texts (Hansen 2000).
- NP recognition, which recognises NP's on the basis of the tags. For this task, at the moment we are using Abney's chunk parser (Abney 1996), a fast and robust system based on finite-state cascades.
- Parsing proper. The final step of linguistic analysis consists in parsing the NP's identified by the NP recogniser to assign them a semantic interpretation.

This paper describes a preliminary study conducted to assess the potential of unification-based parsers, and in particular the LKB parser, to carry out NP parsing in the project.

2 Parsing with the LKB system

The task of the parser is then taken here to be that of assigning NP's a syntactic and semantic representation. In this paper, we assume that semantic analysis is built compositionally from the syntax. Our goal is to investigate whether a system based on feature structures, as used by several modern linguistic theories (LFG, HPSG, etc.), can support the type of semantic representation adopted in the project. The parser chosen for our experiment is called LKB, which stands for Linguistic Knowledge Building. LKB is a grammar and lexicon development system developed at CSLI in Stanford, and running in Allegro Common Lisp. The system has been designed for use with unification-based linguistic formalisms (Lascarides & Copestake 1999), and has been tested extensively with grammars based on HPSG (Pollard & Sag 1994).

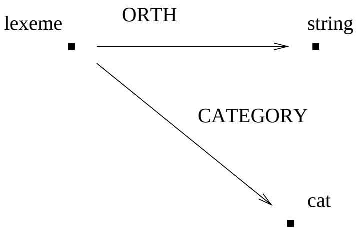

Figure 1: *Graph representation of a feature structure*

A thorough introduction to LKB or even typed features in general is outside the scope of this paper, and the reader is referred to (Copestake 1999) or (Carpenter 1992) on the matter. However, a few concepts need to be explained before delving into a description of our implementation.

First of all, LKB basically operates with one data structure, the typed feature structure, and one operation, unification. In other approaches feature structures are used to augment a context-free backbone. In LKB, on the other hand, everything is made into a typed feature structure, including lexical entries and grammar rules. The types used in an implementation must be organised in a single type hierarchy defining a partial order on the types and specifying which types are consistent.

A typed feature structure can be depicted as a graph where each node corresponds to exactly one type, and where labelled arcs connect the nodes. The labels on the arcs are called attributes. An example is shown in Figure 1. In this graph, the type *lexeme* is represented as having an attribute ORTH whose value is of type *string*, and an attribute CATEGORY whose value is of type *cat*.

A typed feature structure can also be represented as an attribute value matrix (AVM). The example below shows the same feature structure as in Figure 1 in AVM notation.

(1)
$$\begin{bmatrix} \textit{lexeme} \\ \text{ORTH } \textit{string} \\ \text{CATEGORY } \textit{cat} \end{bmatrix}$$

In the LKB representation language, the same typed feature structure is represented in a similar way:

(2)
```
lexeme ::= *top* &
[ ORTH string
  CATEGORY cat ].
```

The type *top* is predefined. It indicates the top of the type hierarchy.

In what follows we will show type definitions in the LKB user language, and output representations in the more widely known AVM notation.

3 A toy NP grammar

The approach to syntactic and semantic analysis adopted here follows HPSG in that we assume a unified feature structure expressing the syntactic structure of the input (in our case NP's) as well as its semantic content. The features chosen to express the semantics, however, have been devised specifically to be easily convertible into the ONTOLOG framework described by Jørgen Fischer Nilsson in this volume.

To illustrate the principles behind our implementation, we will discuss the analysis of the following simple example:

(3) *indhold af betacarotener i fødemidler*
content of betacarotenes in food

The ONTOLOG interpretation of this NP could look as follows:

(4) (indhold $\times$ (CMP: betacaroten) $\times$ (LOC: fødemiddel))

where the concept indhold is related to the concept betacaroten by the relation CMP (comprising), and to the concept fødemiddel by the relation LOC (location). Ontologically, this complex term is a specialisation of the concept (indhold $\times$ (CMP: betacaroten)), which is in turn a specialisation of indhold.

To represent the term in (4) in the form of a feature structure, we define a type *concept_str(ucture)*:

(5)
```
concept_str := sem &
[ CONCEPT concept,
  RELLIST relation_str_list ].
```

This can be read as saying that the feature structure type *concept_str* is a subtype of *sem(antics)*, defined elsewhere as a top type. The type *concept_str* has two attributes, CONCEPT of type *concept*, and RELLIST of type *relation_str_list*. The type *concept* is the topmost type in a hierarchy

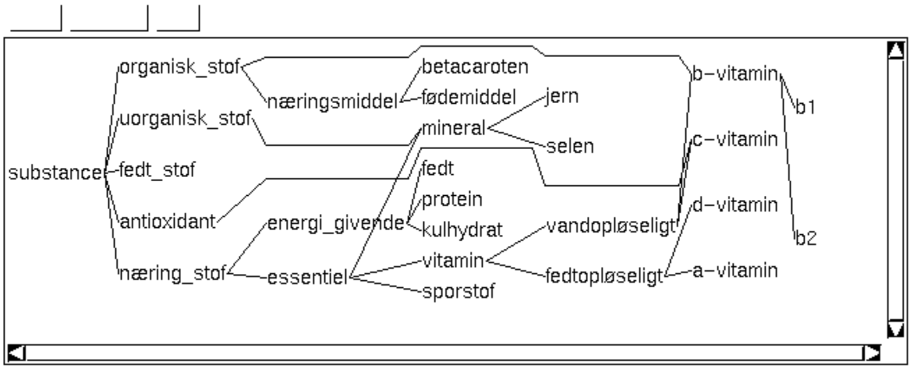

Figure 2: A concept ontology fragment for terms in the nutrition domain

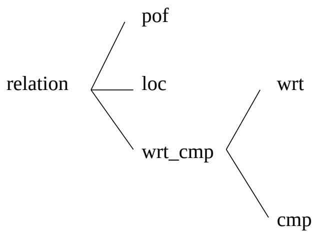

Figure 3: *A simple relation hierarchy*

of concepts a fragment of which is shown in Figure 2[2]. The type *relation_str_list*, on the other hand, is defined as a list of *relation_str(uctures)*, which in turn is defined as shown below:

(6)
```
relation_str := sem &
[ REL relation,
  ARG concept_str ].
```

As can be seen, a *relation_str* has two attributes, a *relation* and a *concept_str*, thus opening up for the necessary recursion. The hierarchy

[2]It will be noted that we mix English and Danish for our concept terms. In general, English is used for more general concepts like substance, and Danish for the more specific ones.

of relations making up the currently very limited definition of the type *relation* is shown in Figure 3.

In addition to these semantic types and attributes, we will also use various syntactic types, such as *lexeme* and *phrase*, and attributes such as CATEGORY and ORTH(ography).

Having set up our types, we need to define the relevant lexicon entries for the words in our example. The entries for indhold and af are shown below in a somewhat simplified form:

(7)
```
indhold := lexeme &
[ ORTH ``indhold'',
  CATEGORY n,
  SEM [ CONCEPT indhold ] ].
```

(8)
```
af := lexeme &
[ ORTH ``af'',
  CATEGORY p,
  SEM [ REL wrt_cmp ] ].
```

The syntactic information is limited here to part-of-speech category (CATEGORY), while an ORTH attribute specifies the string value corresponding to the word form. Furthermore, the SEM attribute contains a specification of the semantics of the two lexemes. In (7), its value is the concept indhold, which places the word in the conceptual hierarchy; in (8), it is the relation WRT_CMP, a supertype of WRT (with respect to) and CMP indicating that the preposition is ambiguous between (at least) two different meanings. Similar lexicon entries are specified for the words *betacarotener*, *i*, and *fødemidler*.

The third component of this simple implementation is of course the grammar, which consists of a number of category specific rules such as the following one[3]:

(9)

```
np_pp_rule := phrase &
[ CATEGORY np,
  SEM [ CONCEPT #concept,
        RELLIST [ FIRST #ppsem,
                  REST #npsem ] ],
```

[3]An alternative strategy is that of using more general schemata expressing dependencies such as that between a head and its complement, or a head and its adjunct as in HPSG.

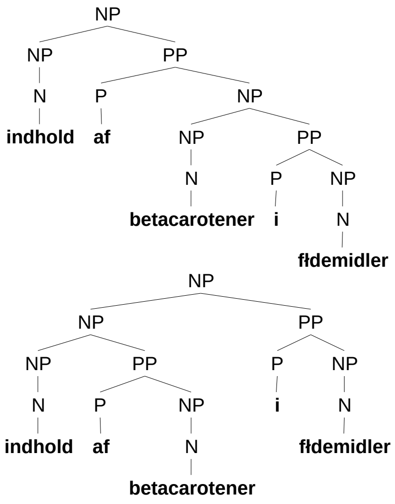

Figure 4: *Phrase structure analyses of the NP "indhold af betacarotener i fødemidler".*

```
ARGS [ FIRST [ CATEGORY np,
               SEM [ CONCEPT #concept,
                     RELLIST #npsem    ] ],
       REST [ FIRST [ CATEGORY pp,
                      SEM #ppsem ],
              REST *null* ] ] ] .
```

The rule describes a feature structure of [CATEGORY *np*], with a list of daughters (ARGS). The head (FIRST) of this list is again of [CATEGORY *np*], and the tail (REST) is of [CATEGORY *pp*]. This corresponds to the context-free rule NP $\rightarrow$ NP PP. Furthermore, the rule states that the CONCEPT attribute of the top NP and that of the embedded NP share the same value (as indicated by the index #concept). The RELLIST of the top, on the other hand, collects its values from the RELLIST of the embedded NP and the SEM of the attached PP (as indicated by the indices #npsem and #ppsem).

With this battery of types, grammar rules and lexicon entries, LKB produces two different analyses of the example under consideration. These are shown in the form of phrase structure trees in Figure 4.

In fact, the way LKB constructs a phrase structure tree is by extracting it from the complex feature structure making up the complete analysis of the input. As already pointed out, syntactic and semantic information is merged in our approach in one and the same feature structure, where the feature SEM is used to represent the semantic content. SEM has different values in the two analyses of our example, as shown below. The interpretation corresponding to the ONTOLOG term in (4) is of course the flatter one in (10.b).

(10.a)
$$\begin{bmatrix} \text{CONCEPT } \textit{indhold} \\ \text{RLIST } \left\langle \begin{bmatrix} \text{REL } \textit{wrt\_cmp} \\ \text{ARG } \begin{bmatrix} \text{CONCEPT } \textit{betacaroten} \\ \text{RLIST } \left\langle \begin{bmatrix} \text{REL } \textit{loc} \\ \text{ARG } \begin{bmatrix} \text{CONCEPT } \textit{f\o demiddel} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

(10.b)
$$\begin{bmatrix} \text{CONCEPT } \textit{indhold} \\ \text{RLIST } \left\langle \begin{matrix} \begin{bmatrix} \text{REL } \textit{wrt\_cmp} \\ \text{ARG } \begin{bmatrix} \text{CONCEPT } \textit{betacaroten} \end{bmatrix} \end{bmatrix}, \\ \begin{bmatrix} \text{REL } \textit{loc} \\ \text{ARG } \begin{bmatrix} \text{CONCEPT } \textit{f\o demiddel} \end{bmatrix} \end{bmatrix} \end{matrix} \right\rangle \end{bmatrix}$$

4 Refining the implementation: semantic selectional restrictions

Compared to the ONTOLOG description in (4), the result shown in (10) is two-way ambiguous. Firstly, the locative PP is taken to modify either the concept indhold or the concept betacaroten. Secondly, the relation expressed by the preposition *af* is analysed as ambiguous between WRT and CMP. The problem of where to attach locative and temporal PPs, which in general can modify more or less all concepts, will not be addressed here. We will, however, build on the implementation described in the preceding section to achieve a more precise analysis of other PP modifiers wherever possible.

Our assumption will be that concepts, in addition to being specified by an *is-a* relation placing them in the ontology they belong to, are also associated with semantic selectional restrictions. These restrictions express the potential semantic contexts in which concepts may occur – contexts where they are related to other concepts by various semantic relations.

For example, we may want to state the fact that the concept indhold is typically related to another concept by the relation CMP (comprising). Of a concept like effect, we may want to say that it potentially takes the relations AGT (agent) and PNT (patient), as in the example below:

(11) *virkning af fedtstofferne på blodet*
(effect of fats on the blood)

To be able to express such semantic selectional restrictions, we redefine our *concept_str* type as follows:

(12)
```
concept_str := sem &
[ ONTO onto,
  RELLIST relation_str_list ].
```

(13)
```
onto := *top* &
[ CONCEPT concept,
  SELECTS relation_str_list ].
```

In this way, the ONTO attribute is used to define a concept together with its selectional restrictions, expressed by the list of relation structures (or semantic roles) which is the value of SELECTS. The attribute RELLIST is still used to record the relations a given concept is actually associated with in a concrete text. Those readers who are familiar with HPSG, may see already at this point the analogy between the SELECTS attribute adopted here and the COMPS list as used in (Pollard & Sag 1994). Both store a list of potential complements; however, the elements on the SELECTS list are semantic roles, while those in the COMPS list are usually expressed in syntactic terms. As we shall see, SELECTS is used in the grammar in a way similar to the COMPS list in HPSG.

Having set up these new types, we can express the necessary restrictions associated with the concepts indhold and effect in the following way:

(14)
```
onto_indhold := onto &
[ CONCEPT indhold,
  SELECTS <cmp_str> ].
```

(15)
```
onto_effect := onto &
[ CONCEPT effect,
  SELECTS <agt_str, pnt_str> ].
```

The expression <cmp_str> means a unary list consisting of the type cmp_str, which expands into:

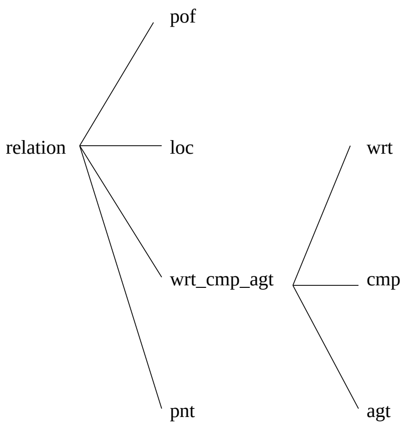

Figure 5: *An augmented relation hierarchy*

(16)
$$\begin{bmatrix} \text{REL } \textit{cmp} \\ \text{ARG} \begin{bmatrix} \text{ONTO} \begin{bmatrix} \text{CONCEPT concept,} \\ \text{SELECTS relation\_str\_list} \end{bmatrix}, \\ \text{RELLIST relation\_str\_list} \end{bmatrix} \end{bmatrix}$$

The expression <agt_str, pnt_str> is a list consisting of two members, an agent relation and a patient relation. They must be added to the typology of relations, which now looks as shown in Figure 5. Furthermore, lexicon entries must be added to our lexicon to account for the words *virkning* ("effect"), *på* ("on"), *fedtstoffer* ("fats"), *blodet* ("the blood"). The REL type of the preposition *af*, which was previously coded as being *wrt_cmp*, must be changed to *wrt_cmp_agt* to accommodate the agentive reading.

The grammar rule in (9) must also be revised. In fact, two rules are now necessary: one to attach PPs that satisfy the head noun's selectional restrictions, and one attaching other types of PPs (e.g. locatives and temporals). The latter will not be much different from the original np_pp_rule. The former is shown below:

(17)

```
np_pp_comp_rule := phrase &
[ CATEGORY np,
```

```
SEM [ ONTO [ CONCEPT #concept,
             SELECTS #rest],
      RELLIST [ FIRST #ppsem,
                REST #npsem]],
ARGS [ FIRST [ CATEGORY np,
               SEM [ ONTO [ CONCEPT #concept,
                            SELECTS [ FIRST #ppsem,
                                      REST #rest ] ],
                     RELLIST #npsem ] ],
       REST [ FIRST [ CATEGORY pp,
                      SEM #ppsem ],
              REST *null* ] ] ] .
```

Note that the PP which is the second argument of the rule must unify with the FIRST element in the SELECTS list of the embedded NP (they share the index #ppsem), and that the value of SELECTS on the top NP becomes the same as the SELECTS list of the embedded NP minus the content of the PP modifying it (only the value of #rest is shared between the SELECTS lists of the two NP's). Again, the reader versed in HPSG will recognise an adaptation of the Subcategorisation Principle, by which the semantic roles associated with a concept are cancelled one by one as they are realised by the complements of the word expressing the concept.

With this apparatus, the system can now choose a more specific interpretation of the relation expressed by the preposition *af* in example (3). In fact, since the concept indhold selects the relation CMP, that relation will now be chosen instead of the more generic WRT_CMP making up the semantics of the preposition. Similarly, the analysis of example (11) will select the specific senses of the prepositions *af* and *på* selected by the head concept, as shown below:

(18) $$\begin{bmatrix} \text{CONCEPT } \textit{effect} \\ \text{RLIST} \left\langle \begin{bmatrix} \text{REL } \textit{pnt} \\ \text{ARG} \begin{bmatrix} \text{CONCEPT } \textit{blod} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{REL } \textit{agt} \\ \text{ARG} \begin{bmatrix} \text{CONCEPT } \textit{fedt\_stof} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

5 Conclusion

To conclude, the implementation described here has shown that a formalism building on typed feature structures as used in the LKB system, provides an acceptable basis for NP parsing in the OntoQuery project. The formalism supports the definition of concept ontologies and semantic selectional restrictions associated with the concepts, in the form of feature structure types. We have shown how such types can interact with lexicon entries and grammar rules to analyse NP's and assign them semantic interpretations that are easily mapped onto ONTOLOG descriptions. The backbone of our analysis has been, in compliance with project goals, the concept ontology. However, we have also shown how a principle originating from a linguistic theory – the Subcategorisation Principle of HPSG – can be recast in semantic terms to be exploited in an ontology-oriented framework.

Our analysis has focused on a limited set of data, and purposely left out a number of issues. One is the problem of ambiguous attachment of locative and temporal PPs. LKB does not provide any immediate solution to the problem either in terms of packing of ambiguous representations or disambiguation by heuristic principles. We have also not dealt with efficiency issues which are of course of crucial importance in a concrete application. Both questions clearly need to be addressed in the future.

References

Abney, S. (1996), Partial parsing via finite-state cascades, *in* 'Proceedings of the ESSLLI'96 Robust Parsing Workshop'. Available from: http://www.sfs.nphil.uni-tuebingen.de/ abney/.

Allan, J. (2000), Natural laguage processing for information retrieval. Tutorial at ANLP/NAACL 2000. Available from: http://ciir.cs.umass.edu/ (last accessed June 2000).

Brill, E. (1995), 'Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging', *Computational Linguistics* **21**(4), 543–565.

Carpenter, B., ed. (1992), *The logic of typed feature structures*, Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, England.

Copestake, A. (1999), *The (new) LKB system – version 5.2*, CSLI, Stanford. Available from: http://www-csli.stanford.edu/ aac/lkb.html.

Hansen, D. H. (2000), Træning og brug af brill-taggeren på danske tekster, Technical report, Center for Sprogteknologi (CST).

Lascarides, A. & Copestake, A. (1999), 'Default representation in constraint-based frameworks', *Computational Linguistics* **25**(1), 55–106.

Pollard, C. & Sag, I. A. (1994), *Head-Driven Phrase Structure Grammar*, Chicago University Press, Chicago.

Srihari, R. & Li, W. (2000), A question answering system supported by information extraction, *in* 'Proceedings of the 6th Applied Natural language Processing Conference and of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics', ACL, Seattle, Washington (USA), pp. 166–172.

Strzalkowski, T., Lin, F. & Perez-Carballo, J. (1998), Natural language information retrieval TREC-6 report, *in* 'Proceedings of TREC-6'. Available from: http://trec.nist.gov/pubs/trec6/t6_proceedings.html (last accessed June 2000).

Voorhees, E. M. & Tice, D. M. (2000), The TREC-8 Question Answering track evaluation, *in* 'Proceedings of TREC-8'. Available from: http://trec.nist.gov/pubs/trec8/t8_proceedings.html (last accessed June 2000).