Understanding Long Sentences

Svetlana Sheremetyeva

LanA Consulting ApS
Mynstersvej 7a, 2
DK-1827, Copenhagen, Denmark

Abstract. This paper describes a natural language understanding component for parsing long sentences. The NLU component includes a generation module so that the results of understanding can be displayed to the user in a natural language and interactively corrected before the final parse is sent to a subsequent module of a particular application. Parsing proper is divided into a phrase level and a level of individual clauses included in a sentence. The output of the parser is an interlingual representation that captures the content of a whole sentence. The load of detecting the sentence clause hierarchy level is shifted to the generator. The methodology is universal in the sense that it could be used for different domains, languages and applications. We illustrate it on the example of parsing a patent claim, - an extreme case of a long sentence.

1 Introduction

NLU systems are numerous, and are based on different approaches depending upon the purpose of an application, using deep or shallow methods of understanding the input content. Shallow understanding can be as simple as spotting keywords. Deeper analysis involves a number of linguistic methods, such as tagging, syntactic parsing and verb dependency relationships. The trade-off between these two approaches is usually one of speed versus the ability to deal with ambiguity of one form or another. NLU performance in applications where deeper than key word understanding is required (e.g., machine translation) drops significantly for sentences containing more than 25-35 words due to a lot of ambiguities that are generated during parsing. For many practical applications a notable improvement of NLU components is demanded specifically for long sentences.

This paper describes a natural language understanding component (NLU) of a multilingual patent workstation that can parse long and syntactically complex sentences[1]. We concentrate on understanding the crucial part of a patent document, - patent claim, - an ultimate example of a long and complex sentence.

The initial task our NLU component is parsing free claim texts, which results in an interlingual representation that captures the content of a whole long claim sentence. This representation is deep enough to provide knowledge for high quality out-

[1] The analyzer we describe is of course applicable to simple and/or short sentences as well.

put for several patent related applications, machine translation being the most important one.

The NLU component is based on empirical data gathered from real patent claims and feature robust data-tuned techniques that allow for natural language understanding of a combination of such linguistic phenomena as long distance dependencies, parallel structures, noun and verb attachments of prepositional phrases, syntactic gaps, etc., known as hard parsing problems. Parsing proper is based on domain-tuned lexicon and a combination of phrase- and dependency grammars.

Although the correlation between the sentence length and ambiguity is clear, the great portion of ambiguities occurs in treatment the higher (clause) nodes in the syntactic tree, on the contrary, processing on the phrase level, such as NP, PP, etc., do not usually generate more ambiguity as a sentence becomes longer [1].

The specificity of our approach is that it does not rely on the structural information of higher levels than a simple clause in the syntactic tree. Our strategy is to divide the analysis into two levels, - a phrase level and a simple clause level, and to use a generator in the NLU component to take over the load of detecting the information about the hierarchy of individual clauses of a long sentence. In addition to this the generation module is used to display the results of clause understanding to the user so that these results can be interactively corrected, if necessary, before the final parse is sent to a subsequent module of a particular application. The generator is equipped with a natural language interface at the user end.

The NLU component has been initially modeled for patent claims in English and Russian but it is readily extensible to other languages, - Danish and Swedish are now being added to the system. For better understanding all examples in this paper are in English. The implementation of the NLU component is done in C++.

2 Related Work

One known strategy for parsing long sentences is to automatically limit the number of words. Very long sentences are broken up along punctuation and certain words (like "which"), and parsed separately as, for example, in many MT systems [2]. This method can lead to errors in the subsequent analysis, as segmentation can be incorrect due to the punctuation ambiguity. The problem is sometimes approached by being very selective about which sentences to parse. To find the most relevant sentences, a statistical relevance filter is built, which returns a relevance score for each sentence in a message [3]. In the Pattern-based English-Korean MT chunking information is used on the phrase-level of the parse result to which a sentence pattern is applied directly thus overcoming many problems in sentence segmentation [4].

A number of successful statistical parsers [5] interleave all disambiguation tasks demonstrating a good performance. It is seldom possible, however, to directly incorporate such parsers into a particular NLP application due to unavailability of tagged corpora for parser retraining. There are parsers that use a constraint-based integration of deep and shallow parsing techniques, e.g., a combination of a shallow tagger or chunker and a deep syntactic parser [6]. There is also a trend to merge different stages of parsing by the use of “supertags”[2] [7] to denote, e.g., elementary trees of Lexicalized Tree-Adjoining Grammar, so that tagging results is almost parsing. Most of computational linguistic research in the patent sublanguage is devoted to information retrieval [8] and use shallow parses tuned to certain technical domains.

3 Data

Analysis of a 9-mio-word corpus of US patents and corpora of available patents in other languages allowed us to incorporate linguistic knowledge and human expertise into our automated NLU system. The data we have collected spans different subject matters[3] and technical domains of invention descriptions. Extremely high structural similarity of claim texts in different national languages gives a unique opportunity to reuse a lot of the English domain lexical/grammar knowledge for other languages. We describe the specificity of patent domain on the example of the English language.

The sublanguage of patent claims is very restricted, - our corpus amounted to approximately 60.000 different lexemes that feature restricted paradigms and a rather strong lexical, morphological and syntactic correlation. The claim consists of a single, albeit possibly very complex, sentence including a large number of clauses that can be over a page long (see Figure 1).

Apparatus for checking dimensions of workpieces ***comprising***
a support structure,
a reference means,
a support means,
said reference and support means ***being coupled*** *to the support structure for supporting and positioning workpieces, coupling locations of reference and support means* ***being*** *basically* ***different****,*
a support frame ***carried*** *by the support structure,*
a plurality of measuring units, and
coupling means for coupling the support frame to the support structure, a plurality *of measuring units* ***being fixed*** *to the support frame for providing corresponding signals* ***processed*** *for performing combined measurements, the coupling means* ***comprises*** *a coupling device which* ***is coupled*** *to the support frame for …*

Fig. 1. A fragment of a US patent claim. Predicates of embedded structures are bold-faced.

The syntactic structure of the claim text is similar across different technical domains and feature similar linguistic restrictions [9] that are hard-coded in lexicon and grammar rules of the NLU component.

[2] The term “supertag” is now used by researches that fill it with a different content. In general a “supertag” means a lexeme label, which codes richer knowledge than a standard POS tag.

[3] Subject matter is a type of invention, such as apparatus, method, process, substance, living organism, etc., as defined by the Patent law.

4 Lexicon

The lexicon contains a list of single sense entries and is built over

Semantic features: SEM_Cl - semantic class, CASE_ROLEs, - a set of case roles associated with a lexeme, if any,

Syntactic features: FILLERs, - sets of most probable fillers of case-roles in terms of types of phrases and lexical preferences,

Linking features: PATTERNs, - linearization patterns of lexemes that code both the knowledge about co-occurrences of lexemes with their case-roles and the knowledge about their linear order (local word order), and

Morphological features: POS, - part of speech, MORPH, - wordforms, number, gender, etc.; the sets of parts of speech and wordforms are domain and application specific.

5 Grammar Formalism

The grammar in our system is a mixture of context free strongly lexicalized Phrase Structure Grammar (PG) and Dependency Grammar (DG) formalisms. The specificity of our grammar is that it is not required to assign a complete syntactic parse to a sentence. In fact a complete syntactic parse is skipped[4] in favor of a deeper sentence representation in terms of semantic dependencies in predicate/case-role structures.

The PG component in our model is built over the domain lexicon and the space of features as specified in the lexicon. It does not use the basic GP rule "S = NP+VP" and is only used on the phrase (NP, PP, etc.) level. In other words, this grammar component covers only those linguistic entities that are *neither predicates, nor clauses* in a complex sentence. We thus do not consider it the task of this level of analysis to give any description of syntactic dependencies.

The grammar rules are domain-tuned rewriting rules augmented with local information, such as lexical preference, and some of rhetorical knowledge, - the knowledge about phrase boundaries, anchored to tabulations, commas and a period (there can only be one rhetorically meaningful period in a claim). The rules cover phrases with enumerations. For example, the grammar will recognize such a phrase as "*several rotating, spinning and twisting elements*" as NP. The head of a phrase (its most important lexical item) is described by a special set of rules.

The second component of our grammar is a strongly lexicalized Case-Role Dependency Grammar [10]. All knowledge within this grammar is anchored to one type of lexemes, namely *predicates* (normally verbs). This grammar component is specified over the space of phrases (NP, PP, etc.) and predicates with all their features as specified in the lexicon.

The analysis grammar assigns the sentence an interlingual representation in the form shown in Figure 2.

[4] This does not mean that the grammar cannot assign a syntactic parse. It is simply not needed in our approach.

```
Sentence::={ proposition){proposition}*
proposition::={label predicate-class predicate ((case-role)(case-role)*}
case-role::= (rank status value)
value::= phrase{(phrase(word supertag)*)}*,
```

Fig. 2. An interlingual representation of a claim sentence, where *label* is a unique identifier of the elementary predicate-argument structure (by convention, marked by the number of its predicate as it appears in the claim sentence, *predicate-class* is a label of an ontological concept, *predicate* is a string corresponding to a predicate from the system lexicon, *case-roles* are *ranked* according to the frequency of their cooccurrence with each predicate in the training corpus, *status* is a semantic status of a case-role, such as agent, theme, place, instrument, etc., and *value* is a string which fills a case-role. *Supertag* is a tag, which conveys both morphological information and semantic knowledge as specified in the shallow lexicon. *Word* and *phrase* are a word and phrase (NPs, PPs, etc.) as specified by PG grammar.

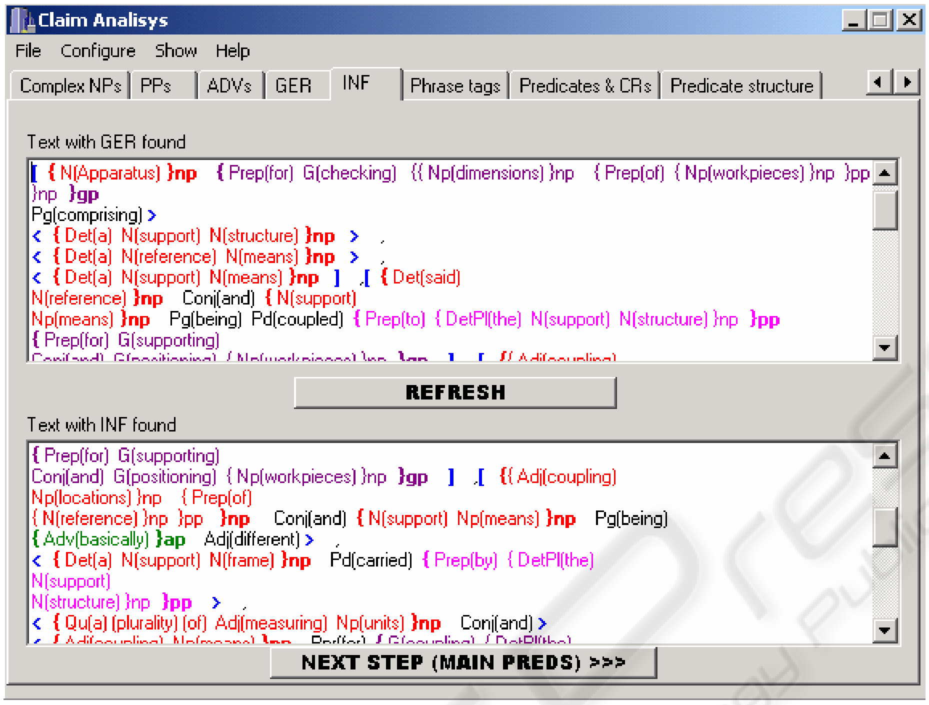
6 Parsing

The parsing module consists of a supertagger[5], chunker and deep semantico-syntactic parser. Our chunker does not only identify coherent word sequences as most typical chunkers but also detects the internal structure of chunks.

The parsing algorithm interleaves semantic, syntactic and morphological knowledge at all levels of analysis, - it uses semantic knowledge from the lexicon for both morphological and syntactic disambiguation, while the morphological and syntactic knowledge is used, in turn, for semantic disambiguation of predicates and case-roles. For selection from among a set of unresolved alternatives licensed by hand-built grammar rules we use data-driven heuristics.

Parsing is done bottom up. The parse is pursued "best first" decision according to a set of *heuristics* compiled through lots of experience parsing. We assume that parse trees are not built by the grammar, but rather are the responsibility of the parser. The result of the parser will thus be the best of all possible parse trees rather than an enumeration of all parse trees.

[5] A supertagger is a tagger that assigns a label coding morphological, syntactic, semantic and linking features of a wordform as specified in the NLU lexicon. We thus use Joshi's term in a different meaning.


Claim Analisys
File Configure Show Help
Complex NPs | PPs | ADVs | GER | INF | Phrase tags | Predicates & CRs | Predicate structure
REFRESH
NEXT STEP (MAIN PREDS) >>>

Text with GER found

```
[ { N(Apparatus) }np  { Prep(for) G(checking)  {{ Np(dimensions) }np  { Prep(of) { Np(workpieces) }np }pp
}np }gp
Pg(comprising) >
< { Det(a) N(support) N(structure) }np  >  ,
< { Det(a) N(reference) N(means) }np  >  ,
< { Det(a) N(support) N(means) }np  ]  ,[ { Det(said)
N(reference) }np  Conj(and) { N(support)
Np(means) }np  Pg(being) Pd(coupled) { Prep(to) { DetPl(the) N(support) N(structure) }np }pp
{ Prep(for) G(supporting)
```

Text with INF found

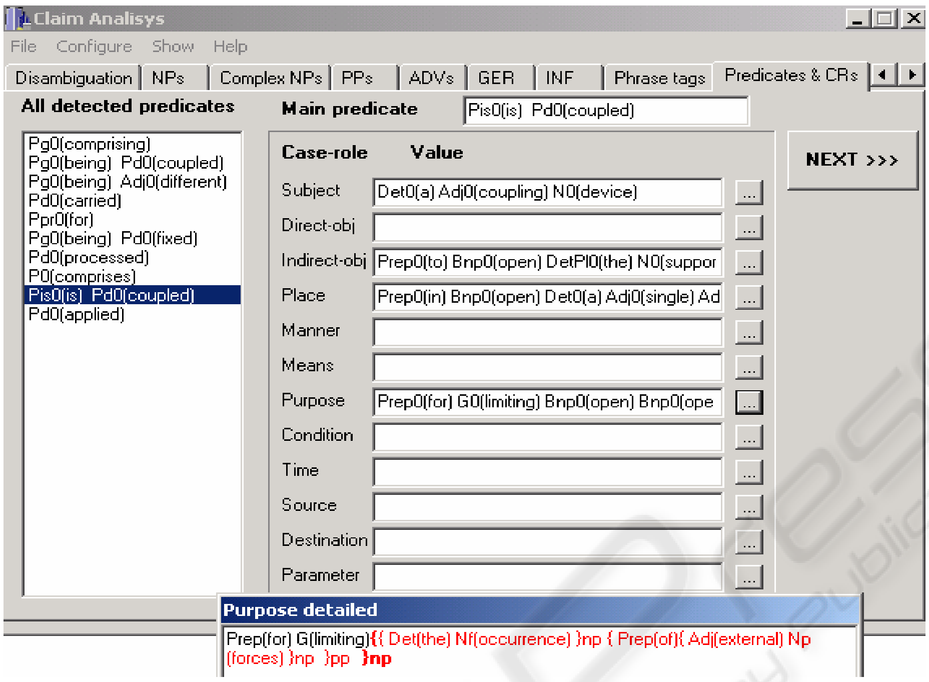
```
{ Prep(for) G(supporting)
Conj(and) G(positioning) { Np(workpieces) }np }gp  ]  ,[  {{ Adj(coupling)
Np(locations) }np  { Prep(of)
{ N(reference) }np }pp  }np   Conj(and) { N(support) Np(means) }np   Pg(being)
{ Adv(basically) }ap   Adj(different) >  ,
< { Det(a) N(support) N(frame) }np   Pd(carried) { Prep(by) { DetPl(the)
N(support)
N(structure) }np  }pp  >  ,
< { Qu(a) (plurality) (of) Adj(measuring) Np(units) }np   Conj(and) >
```

Fig. 3. A screen shot of the interface of the developer tool with traces of chunking noun, prepositional, adverbial, gerundial and infinitival phrases.

The basic parsing scenario consists of the following sequence of procedures:

- *Tokenization*
- *Supertagging*
- *Supertag disambiguation*
- *Chunking of*
    - *Simple noun phrases (NPs)*
    - *Complex noun phrases(complex NPs)*
    - *Prepositional phrases (PPs)*
    - *Adverbial phrases(AdvPs)*
    - *Gerundial Phrases (GerPs)*
    - *Infinitival Phrases(InfPs)*
- *Identifying predicates*
- *Assigning case-roles*
- *Predicate disambiguation*
- *Case-role correction*

Tokenization detects tabulation and punctuation assigning them different types of "boundary" tags. Unlike many other parsers our component does not process segments between the boundary tags. These tags as extra features are used to augment the resolution power of disambiguation rules.



All detected predicates

Pg0(comprising)
Pg0(being) Pd0(coupled)
Pg0(being) Adj0(different)
Pd0(carried)
Ppr0(for)
Pg0(being) Pd0(fixed)
Pd0(processed)
P0(comprises)
Pis0(is) Pd0(coupled)
Pd0(applied)

Main predicate: Pis0(is) Pd0(coupled)

Case-role	Value
Subject	Det0(a) Adj0(coupling) N0(device)
Direct-obj	
Indirect-obj	Prep0(to) Bnp0(open) DetPl0(the) N0(suppor
Place	Prep0(in) Bnp0(open) Det0(a) Adj0(single) Ad
Manner	
Means	
Purpose	Prep0(for) G0(limiting) Bnp0(open) Bnp0(ope
Condition	
Time	
Source	
Destination	
Parameter	

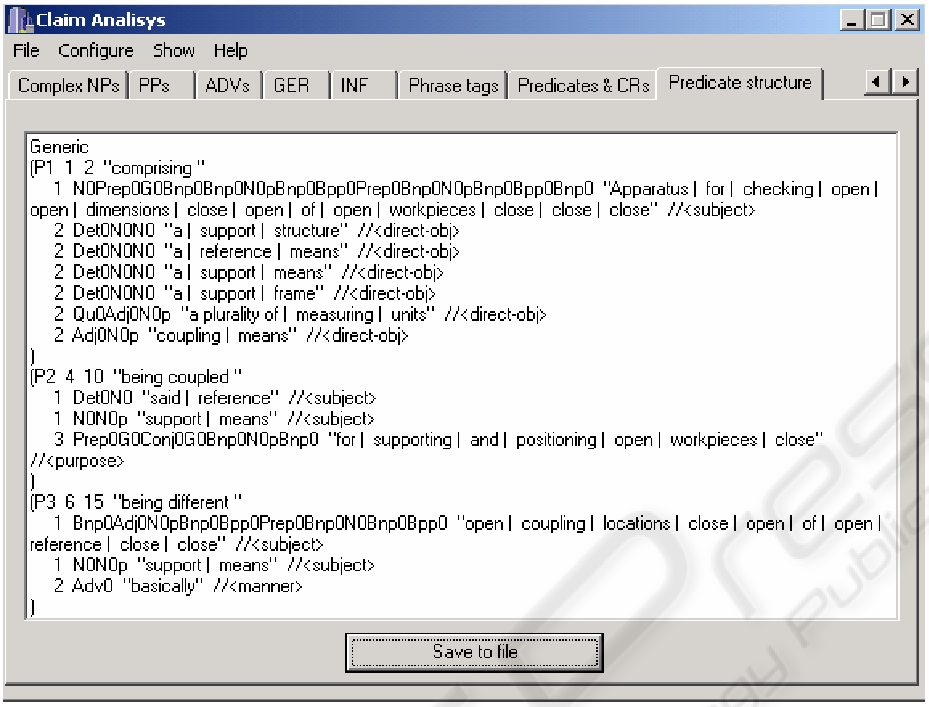
Purpose detailed

Prep(for) G(limiting){{ Det(the) Nf(occurrence) }np { Prep(of){ Adj(external) Np (forces) }np }pp }np

Fig. 4. A screenshot of the developer tool interface with traces of detecting predicates/case-roles for the sentence shown in Figure 1.

Supertagging generates all possible assignments of supertags to words. The ***supertag disambiguation*** procedure uses constraint-based domain specific rules in a 5-word window context with the supertag in question in the middle. The conditioning information includes lexical preferences, “supertags” and “boundary” tags. The disambiguation rules are both “reductionistic” and “substitution” ones. If there are still ambiguities pending after this step of disambiguation the program outputs the most frequent reading in a multiple supertag.

The ***chunking*** procedure is carried out by matching the strings of supertags against patterns in the right hand side of the rules in the PG component of our grammar. The chunking procedure is a succession of processing steps which starts with the detection of simple NPs, followed by the detection of complex NPs, which integrates simple noun phrases into more complex structures (those including prepositions and conjunctions). Due to the rich feature set the chunker can disambiguate such complex NPs as “coupling locations of reference and support means”. After complex NPs boundaries are placed the PPs, AdvPs, GerPs, and, finally, InfPs are chunked in turn. The ordering is based on a set of heuristics. The traces of chunking of the example claim are shown in Figure 3. The next parsing step is the procedure ***identifying predicates.*** At this step in addition to PG we start using our DG mechanism and knowledge stored in the lexicon. This procedure searches for all possible proposition predicates over the “residue” of “free” supertagged words in a chunked sentence and returns predicates of the nascent predicate/case-role structures.


Claim Analisys
File Configure Show Help
Complex NPs PPs ADVs GER INF Phrase tags Predicates & CRs Predicate structure
Save to file

```
Generic
(P1  1  2 "comprising "
    1  N0Prep0G0Bnp0Bnp0N0pBnp0Bpp0Prep0Bnp0N0pBnp0Bpp0Bnp0 "Apparatus | for | checking | open |
open | dimensions | close | open | of | open | workpieces | close | close | close" //<subject>
    2  Det0N0N0 "a | support | structure" //<direct-obj>
    2  Det0N0N0 "a | reference | means" //<direct-obj>
    2  Det0N0N0 "a | support | means" //<direct-obj>
    2  Det0N0N0 "a | support | frame" //<direct-obj>
    2  Qu0Adj0N0p "a plurality of | measuring | units" //<direct-obj>
    2  Adj0N0p "coupling | means" //<direct-obj>
)
(P2  4  10 "being coupled "
    1  Det0N0 "said | reference" //<subject>
    1  N0N0p "support | means" //<subject>
    3  Prep0G0Conj0G0Bnp0N0pBnp0 "for | supporting | and | positioning | open | workpieces | close"
//<purpose>
)
(P3  6  15 "being different "
    1  Bnp0Adj0N0pBnp0Bpp0Prep0Bnp0N0Bnp0Bpp0 "open | coupling | locations | close | open | of | open |
reference | close | close" //<subject>
    1  N0N0p "support | means" //<subject>
    2  Adv0 "basically" //<manner>
)
```

Fig. 5. A screenshot of the developer interface with a fragment of a final parse of the sentence in Figure 1. Fillers of the “direct-obj” case-role are long distance dependencies of the predicate “comprising”. Individual predicate structures code the content of claim clauses; no clause hierarchy is detected so far.

The parser is capable to extract distantly located parts of one predicate, e.g. the predicate “being different” from the fragment, “coupling locations of reference and support means being basically different”**.** We postpone the disambiguation of polysemantic predicates till later.

The ***assigning case-roles*** procedure retrieves semantic dependencies (case-roles of predicates). It detects the governing predicate for every chunked phrase and assigns it a certain case-role status. The rules can use a *5-phrase* context with the phrase in question in the middle.

The conditioning knowledge is very rich at this stage. It includes syntactic and lexical knowledge about phrase constituents, knowledge about supertags and “boundary” tags, and all knowledge from the lexicon. This rich feature space allows for quite a good performance in solving most of the difficult analysis problems such as, recovery of empty syntactic nodes, long distance dependencies, disambiguation of parallel structures and PP attachment without yet disambiguating polysemantic predicates. We do not only try to resolve between noun and verb attachments of PPs, but also between different case-role statuses of PPs within the verb attachment. The relevance of this finer disambiguation for such applications as, e.g., MT is evident; it can affect, for example, the order of realization of PPs in the translation. We attempt to disambiguate case-role statuses that can be assigned to PPs by using heuristics based on lexical and syntactic information from the lexicon. The result of this stage of analysis for the predicate *"coupled"* is shown in Figure 4.

In general, at this stage there can sometimes be several matches between a set of case-roles associated with a particular phrase within one predicate structure and other mistakes which to a great extend can be corrected with heuristics based on the probabilistic knowledge about case-role weights from the lexicon given the meaning of a predicate.

The procedure ***predicate disambiguation*** runs using all the static and dynamic knowledge collected so far. It starts with matching the set of case-roles of a polysemantic predicate identified in the sentence against those present in all omonymous predicate entries. A special metrics is developed to make disambiguation decisions. Once a predicate is disambiguated the procedure ***correct case-role*** attempts to correct a case-role status of a phrase if it does not fit the predicate description in the lexicon.

Figure 5 shows a fragment of the final parse for our example. Individual predicate structures code the content of claim clauses. Strictly speaking this parse is partial as no clause hierarchy is detected so far. However, we only need clause hierarchy knowledge for the applications that include generation of a patent claim in a legal format, for example, for machine translation.

The generator, as mentioned earlier, makes up for the incomplete parse of the analyzer, - it specifies the hierarchy of claim clauses by "gluing" the individual predicate structures into a forest of trees and linearising the trees bypassing them in a specified order. The tree building and tree bypassing algorithms are universal for any language. They draw on legal knowledge about patents and predicate features as specified in the lexicon. Linearised trees of predicate structures include complete information about clause hierarchy.

7 Conclusions and Future Work

We presented the grammar-based data-intensive NLU component for patent claims, which is now implemented in its demo version. Preliminary results show a reasonably small number of failures, mainly due to the incompleteness of analysis rules and lexicon. The final parse can be used in any patent related application. We will sketch just three of them, - machine translation, improving readability of patent claims and information retrieval.

A machine translation system APTrans for the English-Danish language pair is currently under development. The main simplification here is to use the NLU parse representation as transfer invariant thus reducing translation process to only translating case-role fillers. A parse structure with TL fillers is then input into a generation module, which due to cross-linguistic structural similarity of claims mostly reuses fully operational English claim generator AutoPat [11].

An application for improving readability of patent claims decomposes a complex claim sentence into a set of simple sentences. It is in fact a stand-alone NLU component with a user interface.

Information retrieval and extraction of patent information based on NLU of patent claims is our perspective application. The task here is to index patent corpora with the parse structures and query stored texts in their internal representation. The query will be given as free text to be analyzed also by the NLU component. A projection of the query representation onto claim text representations will then be performed. When a good match occurs, the text is selected as an answer to the query. Generation of query/question specific information will also be possible from claims internal representation.

References

1. Abney, S.: Part-of-speech Tagging and Partial Parsing. In: Corpus-Based methods in Language and Speech. Klue Academic Publishes (1996).
2. Kim Y., Ehara T.: A Method for Partitioning of Long Japanese Sentences with Subject Resolution in J/E Machine Translation. In: Proceedings of the 1994 ICCPOL (1994)
3. Hobbs J. R., Bear J.: Two Principles of Parse Preference. In: Linguistica Computazionale: Current Issues in Computational Linguistics: In Honour of Don Walker, Vol. 9-10 (1995)
4. Roh Y., Hong M., Choi S., Lee K., Park S.: For the Proper Treatment of Long Sentences in a Sentence Pattern-based English-Korean MT System. In; Proceedings of MT Summit IX. New Orleans (2003)
5. Charniak E.: A Maximum-entropy-inspired Parser. Proceedings of the North American Chapter of the Association of Computational Linguistics (2000)
6. Daum M., Killian A.F., Menzel W.: Constraint based Integration of Deep and Sallow Parsing Techniques. In: Proceedings of the European Chapter of the Association of Computational Linguistics, Budapest (2003)
7. Joshi A., Srinivas B.: Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. http://acl.ldc.upenn.edu/C/C94/C94-1024.pdf. (1994)
8. Fujii A., Ishikawa T.: NTCIR-3 Patent Retrieval Experiments at ULIS. In Proceedings of the Third NTRCIR Workshop (2002)
9. Sheremetyeva, S.: A Flexible Approach To Multi-Lingual Knowledge Acquisition For NLG. In: Proceedings of the 7th European Workshop on Natural Language Generation. Toulouse. (1999)
10. Charles J. Fillmore. 1970. Subjects, speakers and roles. Synthese.21/3/4 (1970)
11. Sheremetyeva S: Embedding MT for Generating Patent Claims in English from a Multilingual Interface. In: Proceedings of the Workshop on Patent Translation in Conjunction with MT Summit, Phuket (2005)