Natural Language Analysis of Patent Claims

Svetlana Sheremetyeva

Department of Computational Linguistics
Copenhagen Business School,
Bernhard Bangs Alle 17 B,
DK-2000, Denmark
lanaconsult@mail.dk

Abstract

We propose a NLP methodology for analyzing patent claims that combines symbolic grammar formalisms with data-intensive methods while enhancing analysis robustness. The output of our analyzer is a shallow interlingual representation that captures both the structure and content of a claim text. The methodology can be used in any patent-related application, such as machine translation, improving readability of patent claims, information retrieval, extraction, summarization, generation, etc. The methodology should be universal in the sense that it could be applied to any language, other parts of patent documentation and text as such.

1 Introduction

An exploding volume of patent applications makes essential the use of adequate patent processing tools that could provide for better results in any field of patent related activity. NLP techniques associated with specificity of patent domain have promise for improving the quality of patent document processing.

Though it is generally recognized that the patent domain features overwhelmingly long and complex sentences and peculiar style (Kando, 2000) only a few researchers really rely on the linguistic specificity of patent style (vs. technical style) when processing patent documentation (Shnimory et al., 2002; Gnasa and Woch, 2002; Fujii and Ishikawa, 2002).

Developing natural language analyzers for patents (with at least one or any combination of morphological, syntactic and semantic modules) is a basic task. The ultimate task of such analysis is to build a kind of possibly unambiguous content representation that could further be used to produce higher quality applications.

Broad coverage syntactic parsers with good performance have recently become available (Charniak, 2000; Collins, 2000), but they are not trained for patents. Semantic parsing is considerably less developed and shows a trend to rely on ontologies rather then semantic primitives. (Gnasa and Woch, 2002).

This paper reports on on-going project whose goal is to propose a NLP methodology and an analyzer for patent claims. The claim is the focal point of a patent disclosure, - it describes essential features of the invention and is the actual subject of legal protection.

The methodology we suggest combines symbolic grammar formalisms with data-intensive knowledge while enhancing analysis robustness. The output of our analyzer is a shallow interlingual representation that captures both the structure and content of a claim text. It can be used in any patent-related application, such as machine translation improving readability of patent claims, information retrieval, extraction, summarization, generation, etc. The methodology should be universal in the sense that it could be applied to any language, other parts of patent documentation and text as such.

In what follows we first consider the knowledge base of our model describing in turn a flexible depth lexicon, grammar formalism, and language of knowledge representation for the final parse. We then focus on the analysis algorithm as a multicomponent procedure. To illustrate the potential of the methodology we further sketch two of its possible applications, namely, machine translation and an application for improving the readability of patent claims. We conclude with the description of the project status and future work.

2 Knowledge

The structure and content of the knowledge base has been designed to a) help solve analysis problems, — different kinds of ambiguity, — and b) minimize the knowledge acquisition effort by drawing heavily on the patent *claim* linguistic restrictions.

A patent claim shares technical terminology with the rest of a patent but differs greatly in its content and syntax. It must be formulated according to a set of precise syntactic, lexical and stylistic guidelines as specified by the German Patent Office at the turn of the last century and commonly accepted in the U.S., Japan, and other countries. The claim describes essential features of the invention in the obligatory form of a single extended nominal sentence, which frequently includes long and telescopically embedded predicate phrases. A US patent claim that we will further use as an example in our description is shown in Figure 1.

A cassette for holding excess lengths of light waveguides in a splice area comprising a cover part and a pot-shaped bottom part having a bottom disk and a rim extending perpendicular to said bottom disk, said cover and bottom parts are superimposed to enclose jointly an area forming a magazine for excess lengths of light waveguides, said cover part being rotatable in said bottom part, two guide slots formed in said cover part, said slots being approximately radially directed, guide members disposed on said cover part, a splice holder mounted on said cover part to form a rotatable splice holder.

Figure 1. A US patent claim text.

In our system the knowledge is coded in the system lexicon, which has been acquired from two kinds of corpora, - a corpus of complete patent disclosures and a corpus of patent claims. The lexicon consists of two parts: *a shallow lexicon* of lexical units and *a deep (information-rich) lexicon* of predicates. Predicates in our model are words, which are used to describe interrelations between the elements of invention. They are mainly verbs, but can also be adjectives or prepositions.

2.1 Shallow Lexicon

The word list for this lexicon was automatically acquired from a 5 million-word corpus of a US patent web site. A semi-automatic supertagging procedure was used to label these lexemes with their supertags.

Supertagging is a process of tagging lexemes with labels (or supertags), which code richer information than standard POS tags. The use of supertags, as noted in (Joshi and Srinivas, 1994) localizes some crucial linguistic dependencies, and thus show significant performance gains. The content of a supertag differs from work to work and is tailored for the needs of an application. For example, Joshi and Srinivas (1994) who seem to coin this term use elementary trees of Lexicalized Tree-Adjoining Grammar for supertagging lexical items. In (Gnasa and Woch, 2002) it is grammatical structures of the ontology that are used as supertags.

In our model a supertag codes morphological information (such as POS and inflection type) and semantic information, an ontological concept, defining a word membership in a certain semantic class (such as object, process, substance, etc.). For example, the supertag Nf shows that a word is a noun in singular (N), means a process (f), and does not end in *–ing.* This supertag will be assigned, for example, to such words as *activation* or *alignment.* At present we use 23 supertags that are combinations of 1 to 4 features out of a set of 19 semantic, morphological and syntactic features for 14 parts of speech. For example, the feature structure of noun supertags is as follows:

```
Tag [ POS[Noun
          [object   [plural, singular]
           process [-ing, other[plural, singular]]
          substance [plural, singular]
          other      [plural, singular]]]]]
```

In this lexicon the number of semantic classes (concepts) is domain based. The “depth” of supertags is specific for every part of speech and codes only that amount of the knowledge that is believed to be sufficient for our analysis procedure.

That means that we do not assign equally "deep" supertags for every word in this lexicon. For example, supertags for verbs include only morphological features such as verb forms (-ing form, -ed form, irregular form, finite form). For finite forms we further code the number feature (plural or singular). Semantic knowledge about verbs is found in the predicate lexicon.

2.2 Predicate Lexicon

This lexicon contains reach and very elaborated linguistic knowledge about claim predicates and covers both the lexical and, crucially for our system, the syntactic and semantic knowledge. Our approach to syntax is, thus, fully lexicalist. Below, as an example, we describe the predicate lexicon for claims on apparatuses. It was manually acquired from the corpus of 1000 US patent claims.

Every entry includes the morphological, semantic and syntactic knowledge.

Morphological knowledge contains a list of practically all forms of a predicate that could only be found in the claim corpus.

Semantic knowledge is coded by associating every predicate with a concept of a domain-tuned ontology and with a set of case-roles. The semantic status of every case-role is defined as "agent", "place", "mode", etc. The distinguishing feature of the case frames in our knowledge base is that within the case frame of every predicate the case roles are ranked according their weight calculated on the basis of the frequency of their occurrence in actual corpus together with the predicate. The set of case-roles is not necessarily the same for every predicate.

Syntactic knowledge includes the knowledge about linearization patterns of predicates that codes both the knowledge about co-occurrences of predicates and case-roles and the knowledge about their liner order in the claim text. Thus, for example, the following phrase from an actual claim: (1: *the splice holder*) *: *is arranged* (3: *on the cover part*) (4: *to form a rotatable splice holder*) (where 1, 3 and 4 are case role ranks and "*" shows the position of the predicate), will match the linearization pattern (1 * 3 4). Not all case-roles defined for a predicate co-occur every time it appears in the claim text. Syntactic knowledge in the predicate dictionary also includes sets of most probable fillers of case-roles in terms of types of phrases and lexical preferences.

2.3 Grammar and Knowledge Representation

In an attempt to bypass weaknesses of different types of grammars the grammar description in our model is a mixture of context free lexicalized Phrase Structure Grammar and Dependency Grammar formalisms.

Our Phrase Structure Grammar consists of a number of rewriting rules and is specified over a space of supertags. The grammar is augmented with local information, such as lexical preference and some of rhetorical knowledge, - the knowledge about claim segments, anchored to tabulations, commas and a period (there can only be one rhetorically meaningful period in a claim which is just one sentence). This allows the description of such phrases as, for example, "*several rotating, spinning and twisting elements*". The head of a phrase (its most important lexical item) is assigned by a grammar rule used to make up this phrase.

The second component of our grammar is a version of Dependency Grammar. It is specified over the space of phrases (NP, PP, etc.) and a residue of "ungrammatical" words, i.e., words that do not satisfy any of the rules of our Phrase Structure Grammar.

The Dependency Grammar in our model is a strongly lexicalized case-role grammar. All syntactic and semantic knowledge within this grammar is anchored to one type of lexemes, namely *predicates* (see Section 2.2). This grammar assigns a final parse (representation) to a claim sentence in the form:

text::={ template){template}*

template::={label predicate-class predicate ((case-role)(case-role)*}

case-role::= (rank status value)

value::= phrase{(phrase(word supertag)*)}*

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 (see Section 2.1). *Word* and *phrase* are a word and phrase (NPs, PPs, etc.) in a standard understanding. The representation is thus quite informative and captures to a large extent both morpho-syntactic and semantic properties of the claim.

For some purposes such set of predicate templates can be used as a final claim representation but it is also possible to output a unified representation of a patent claim as a tree of predicate-argument templates.

3 Analysis algorithm

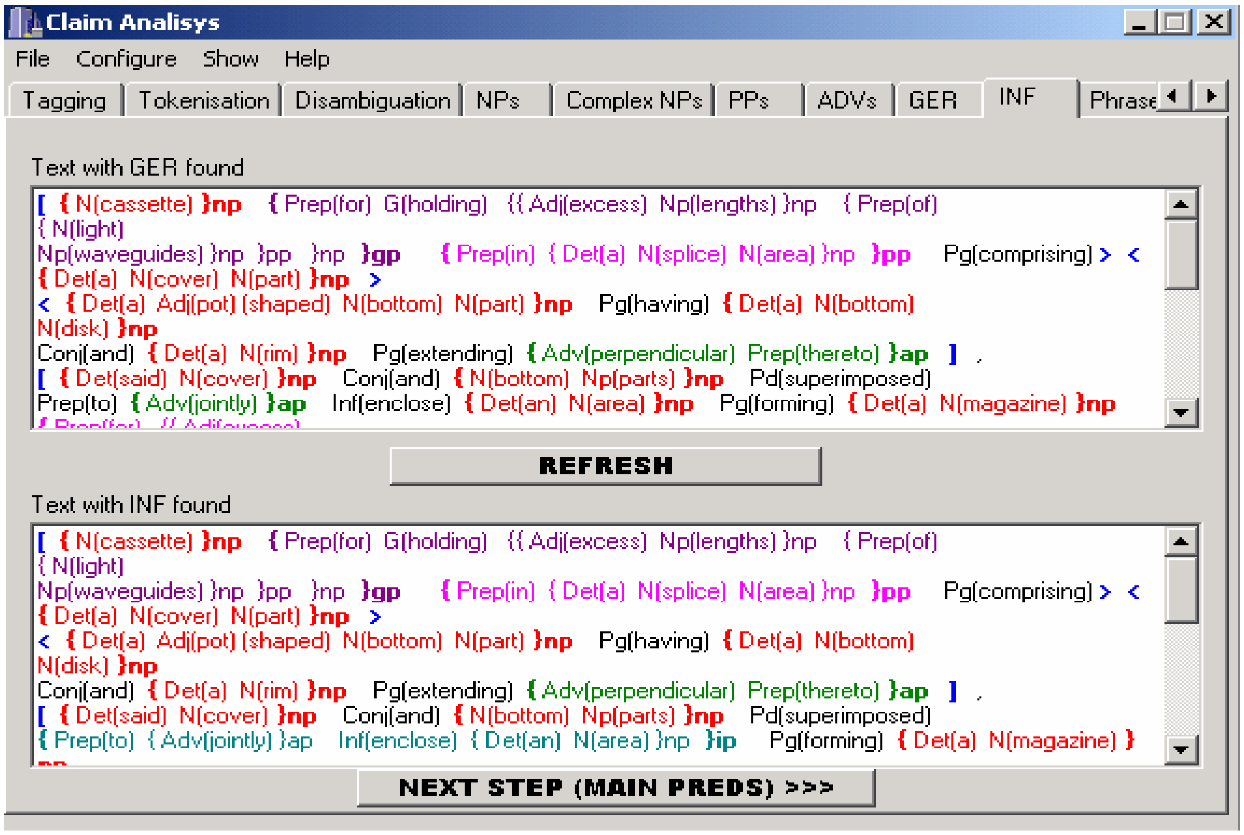
The analyzer takes a claim text as input and after a sequence of analysis procedures produces a set of internal knowledge structures in the form of predicate-argument templates filled with chunked and supertagged natural language strings. The implementation of an experimental version is being carried out in C++. In further description we will use the example of a claim text shown in Figure 1.

The basic analysis scenario for the patent claim consists of the following sequence of procedures:

- *Tokenization*
- *Supertagging*
- *Chunking*
- *Determining dependencies*

Every procedure relies on a certain amount of static knowledge of the model and on the dynamic knowledge collected by the previous analyzing procedures.

The top-level procedure of the claim analyser is ***tokenization***. It detects tabulation and punctuation flagging them with different types of "border" tags. Following that runs the ***supertagging*** procedure, - a look-up of words in the shallow



```
[ { N(cassette) }np   { Prep(for)  G(holding)  {{ Adj(excess)  Np(lengths) }np   { Prep(of)
{ N(light)
Np(waveguides) }np  }pp   }np  }gp    { Prep(in)  { Det(a)  N(splice)  N(area) }np  }pp    Pg(comprising) > <
{ Det(a)  N(cover)  N(part) }np  >
<  { Det(a)  Adj(pot) (shaped)  N(bottom)  N(part) }np   Pg(having)  { Det(a)  N(bottom)
N(disk) }np
Conj(and)  { Det(a)  N(rim) }np   Pg(extending)  { Adv(perpendicular)  Prep(thereto) }ap   ]  ,
[  { Det(said)  N(cover) }np   Conj(and)  { N(bottom)  Np(parts) }np   Pd(superimposed)
Prep(to)  { Adv(jointly) }ap   Inf(enclose)  { Det(an)  N(area) }np   Pg(forming)  { Det(a)  N(magazine) }np
{ Prep(for)  {{ Adj(excess)
```

```
[ { N(cassette) }np   { Prep(for)  G(holding)  {{ Adj(excess)  Np(lengths) }np   { Prep(of)
{ N(light)
Np(waveguides) }np  }pp   }np  }gp    { Prep(in)  { Det(a)  N(splice)  N(area) }np  }pp    Pg(comprising) > <
{ Det(a)  N(cover)  N(part) }np  >
<  { Det(a)  Adj(pot) (shaped)  N(bottom)  N(part) }np   Pg(having)  { Det(a)  N(bottom)
N(disk) }np
Conj(and)  { Det(a)  N(rim) }np   Pg(extending)  { Adv(perpendicular)  Prep(thereto) }ap   ]  ,
[  { Det(said)  N(cover) }np   Conj(and)  { N(bottom)  Np(parts) }np   Pd(superimposed)
{ Prep(to)  { Adv(jointly) }ap   Inf(enclose)  { Det(an)  N(area) }np  }ip    Pg(forming)  { Det(a)  N(magazine) }
```

Figure 2. A screenshot of the developer tool interface, which shows traces of chunking noun, prepositional, adverbial, gerundial and infinitival phrases in the claim text shown in Figure 1.

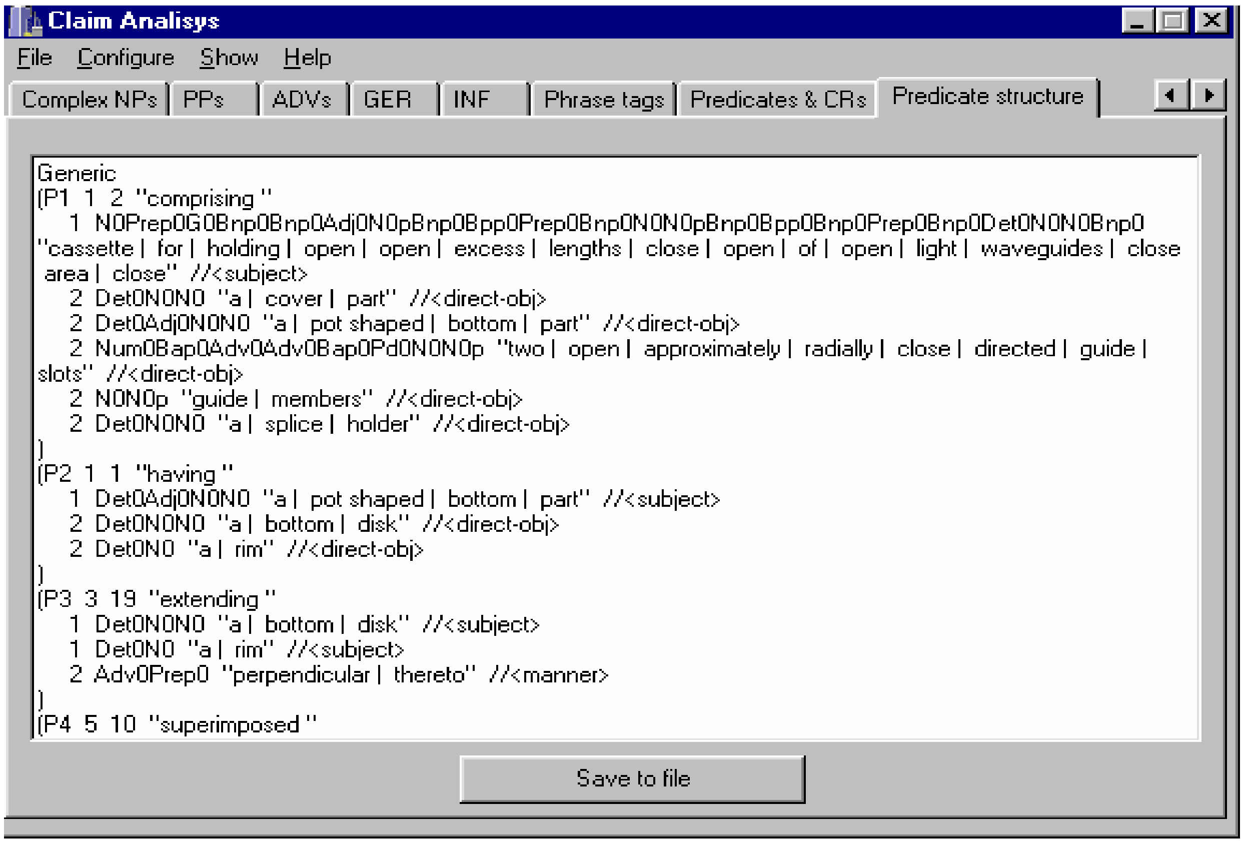
lexicon (see Section 2.1). It generates all possible assignments of supertags to words.

Then the *supertag disambiguation* procedure attempts to disambiguate multiple supertags. It uses constraint-based hand-crafted rules to eliminate impossible supertags for a given word in a 5-word window context with the supertag in question in the middle. The rules use both lexical, "supertag" and "border" tags knowledge about the context. The disambiguation rules are of several types, not only "reductionistic" ones. For example, substitution rules may change the tag "Present Plural" into "Infinitive" (We do not have the "Infinitive" feature in the supertag feature space). If there are still ambiguities pending after this step of disambiguation the program outputs the most frequent reading in the multiple supertag.

After the supertags are disambiguated the ***chunking*** procedure switches on. Chunking 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. "Border" tags are included in the conditioning knowledge.

During the chunking procedure we use only a subset of PG rewriting rules. This subset includes neither the basic rule "S = NP+VP", nor any rules for rewriting VP. This means that at this stage of analysis we cover only those sentence *components that are not predicates of any clause* (be it a main clause or a subordinate/relative clause). We thus do not consider it the task of the chunking procedure to give any description of syntactic dependencies.

The *chunking* procedure is a succession of processing steps itself starting with the *simple-noun-phrase* procedure, followed the *complex-noun-phrase* procedure, which integrates simple noun phrases into more complex structures (those including prepositions and conjunctions). Then the *prepositional-, adverbial-, infinitival- and gerundial-phrase* procedures switch on in turn.



```
Generic
(P1  1  2  "comprising "
    1  N0Prep0G0Bnp0Bnp0Adj0N0pBnp0Bpp0Prep0Bnp0N0N0pBnp0Bpp0Bnp0Prep0Bnp0Det0N0N0Bnp0
"cassette |  for |  holding |  open |  open |  excess |  lengths |  close |  open |  of |  open |  light |  waveguides |  close
 area |  close"  //<subject>
    2  Det0N0N0  "a |  cover |  part"  //<direct-obj>
    2  Det0Adj0N0N0  "a |  pot shaped |  bottom |  part"  //<direct-obj>
    2  Num0Bap0Adv0Adv0Bap0Pd0N0N0p  "two |  open |  approximately |  radially |  close |  directed |  guide |
slots"  //<direct-obj>
    2  N0N0p  "guide |  members"  //<direct-obj>
    2  Det0N0N0  "a |  splice |  holder"  //<direct-obj>
)
(P2  1  1  "having "
    1  Det0Adj0N0N0  "a |  pot shaped |  bottom |  part"  //<subject>
    2  Det0N0N0  "a |  bottom |  disk"  //<direct-obj>
    2  Det0N0  "a |  rim"  //<direct-obj>
)
(P3  3  19  "extending "
    1  Det0N0N0  "a |  bottom |  disk"  //<subject>
    1  Det0N0  "a |  rim"  //<subject>
    2  Adv0Prep0  "perpendicular |  thereto"  //<manner>
)
(P4  5  10  "superimposed "
```

Figure 3. A fragment of the final parse of the sentence in Figure 1. Fillers of the "direct-obj" case-role are long distance dependencies of the predicate "comprising".

The order of the calls to these component procedures in the chunking algorithm is established to minimize the processing time and effort. The ordering is based on a set of heuristics, such as the following. Noun phrases are chunked first as they are the most frequent types of phrases and many other phrases build around them. Figure 1 is a screenshot of the interface of the analysis grammar acquisition tool. It shows traces of chunking noun, prepositional, adverbial, gerundial and infinitival phrases in the example of a claim text shown in the left pane of Figure 3.

The next step in claim analysis is the procedure ***determining dependencies.*** At this step in addition to PG we start using our DG mechanism. The procedure *determining dependencies* falls into two components: *determining elementary (one predicate) predicate-argument structures* and unifying these *structures into a tree*. In this paper we'll limit ourselves to a detailed description of the first of these tasks.

The *elementary predicate structure* procedure, in turn, consists of three components, which are described below.

The fist *find-predicate* component searches for all possible predicate-pattern matches over the "residue" of "free" words in a chunked claim and returns flagged predicates of elementary predicate-argument structures. The analyzer is capable to extract distantly located parts of one predicate (e.g. "*is arranged*" from "*A is substantially vertically arranged on B*").

The second *find-case-roles* component retrieves semantic (case-roles) and syntactic dependencies (such as syntactic subject), requiring that all and only dependent elements (chunked phrases in our case) be present within the same predicate structure.

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 "border" tags, and all the knowledge about the properties of a predicate as specified in the predicate dictionary.

This rich feature space allows quite a good performance in solving the most difficult analysis problems such as, recovery of empty syntactic nodes, long distance dependencies, disambiguation of PP attachment and parallel structures. There can several matches between the set of case-roles associated with a particular phrase within one predicate structure. This type of ambiguity can be resolved with the probabilistic knowledge about case-role weights from the predicate dictionary given the meaning of a predicate.

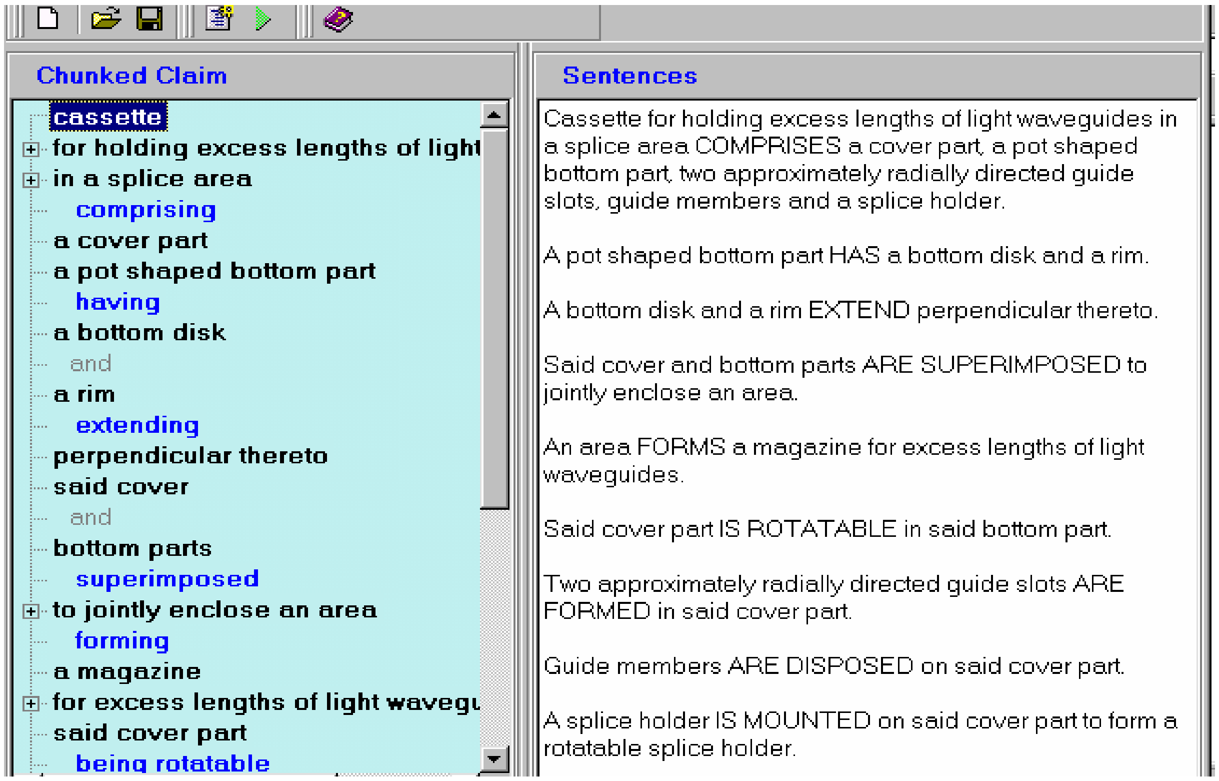
If a predicate is has several meanings then the procedure *disambiguate predicate* starts, which relies on all the static and dynamic knowledge collected so far. During this procedure, once a predicate is disambiguated it is possible to correct a case-role status of a phrase if it does not fit the predicate description in the lexicon.

Figure 3 shows the result of assigning case-roles to the predicates of the claim in Figure 1. The set of predicate-arguments structures conforms the format of knowledge representation given in Section 2.3. As we have already mentioned the analyzer might stop at this point. It can also proceed further and unify this set of predicate structures into a tree. We do not describe this rather complex procedure here and note only that for this purpose we can reuse the planning component of the generator described in (Sheremetyeva and Nirenburg, 1996).

4 Examples of possible applications

In general, the final parse in the format shown in Figure 3 can be used in any patent related application. It is impossible to give a detailed description of these applications in one paper. We thus limit ourselves to sketching just two of them, - machine translation and improving the readability of patent claims.

Long and complex sentences, of which patent claims are an ultimate example, are often mentioned as sentences of extremely low translatability (Gdaniec, 1994). One strategy currently used to cope with the problem in the MT frame is to automatically limit the number of words in a sentence by cutting it into segments on the basis of the punctuation only. In general this results in too few phrase boundaries (and some incorrect ones, e.g. enumerations). Another well-known strategy is pre-editing and postediting or/ and using controlled language, which can be problematic for the MT user. It is difficult to judge
whether current MT systems use more sophisticated parsing strategies to deal with the problems caused by the length and complexity of



Chunked Claim

cassette
for holding excess lengths of light
in a splice area
comprising
a cover part
a pot shaped bottom part
having
a bottom disk
and
a rim
extending
perpendicular thereto
said cover
and
bottom parts
superimposed
to jointly enclose an area
forming
a magazine
for excess lengths of light wavegu
said cover part
being rotatable

Sentences

Cassette for holding excess lengths of light waveguides in a splice area COMPRISES a cover part, a pot shaped bottom part, two approximately radially directed guide slots, guide members and a splice holder.

A pot shaped bottom part HAS a bottom disk and a rim.

A bottom disk and a rim EXTEND perpendicular thereto.

Said cover and bottom parts ARE SUPERIMPOSED to jointly enclose an area.

An area FORMS a magazine for excess lengths of light waveguides.

Said cover part IS ROTATABLE in said bottom part.

Two approximately radially directed guide slots ARE FORMED in said cover part.

Guide members ARE DISPOSED on said cover part.

A splice holder IS MOUNTED on said cover part to form a rotatable splice holder.

Figure 4. A screenshot of the user interface of a prototype application for improving the readability of patent claims. The right pane shows an input claim (see Figure 1) chunked into predicates and other phrases (case-role fillers). The structure of complex phrases can be deployed by clicking on the "+" sign. The right pane contains the claim text a set of simple sentences.

of real life utterances as most system descriptions are done on the examples of simple sentences.

To test our analysis module for its applicability for machine translation we used the generation module of our previous application, - AutoPat, - a computer system for authoring patent claims (Sheremetyeva, 2003), and modeled a translation experiment within one (English) language, thus avoiding (for now) transfer problems[1] to better concentrate on the analysis proper. Raw claim sentences were input into the analyzer, and parsed. The parse was input into the AutoPat generator, which due to its architecture output the "translation" in two formats, - as a single sentence, which is required when a claim is supposed to be included in a patent document, and as a set of simple sentences in TL. The modules proved to be compatible and the results of such "translation" showed a reasonably small number of failures, mainly due to the incompleteness of analysis rules.

The second type of the translation output (a set of sentences), shows how to use our analyzer in a separate (unilingual or multilingual) application for improving the readability of patent claims, which is relevant, for example, for information dissemination. Figure 4 is a screenshot of the user interface of a prototype of such an application.

We are aware of two efforts to deal with the problem of claim readability. Shnimory et. al (2002) investigate NLP technologies to improve readability of Japanese patent claims concentrating on rhetorical structure analysis. This approach uses shallow analysis techniques (cue phrases) to segment the claim into more readable parts and visualizes a patent claim in the form of a rhetorical structure tree. This differs from our final output, which seems to be easier to read. Shnimory et. al

[1] The transfer module (currently under development) transfers every individual SL parse structure into an equivalent TL structure keeping the format of its representation. It then "glues" the individual structures into a tree to output translation as one sentence or generates a set of simple sentences directly from the parse in Figure 3.

(cf.) refer to another NLP research in Japan directed towards dependency analysis of patent claims to support analytical reading of patent claims. Unfortunately the author of this paper cannot read in Japanese. We thus cannot judge ourselves how well the latter approach works.

5 Status and Future Work

The analyzer is in the late stages of implementation as of May 2003. The static knowledge sources have been compiled for the domain of patents about apparatuses. The morphological analysis and syntactic chunking are operational and well tested. The case-role dependency detection is being currently tested and updated. The compatibility of the analyzer and fully operational generator has been proved and tested. First experiments have been done to use the analyzer for such applications as machine translation and improving claim readability. We have not yet made a large-scale evaluation of our analysis module. This leaves the comparison between other parsers and our approach as a future work. The preliminary results show a reasonably small number of failures, mainly due to the incompleteness of analysis rules that are being improved and augmented with larger involvement of predicate knowledge.

We intend to a) add an optional interactive module to the analyzer (that would allow for human interference into the process of analysis to improve its quality), and complete the integration of the analyzer into a machine translation system and an application for improving claim readability. Another direction of work is developing applications in a variety of languages (software localization); b) develop a patent search and extraction facility on the basis of the patent sublanguage and our parsing strategy.

References

Akiro Shnimori, Manabu Okumura,Yuzo Marukawa, and Makoto IwaYama. 2002. Rethorical Structure Analysis of Japanese Patent Claims Using Cue Phrases. Proceedings of the Third NTRCIR Workshop.

Aravind K.Joshi and Bangalore Srinivas. 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. http://acl.ldc.upenn.edu/C/C94/C94-1024.pdf

Atstushi Fujii and Tetsuya Ishikawa. 2002. NTCIR-3 Patent Retrieval Experiments at ULIS. *Proceedings of the Third NTRCIR Workshop.*

Claudia Gdaniec. 1994. The Logos Translatability Index in Technology Partnerships for Crossing the Language Barrier. *Proceedings of the First Conference of the Association for Machine Translation in the Americas (AMTA).*

Don Blaheta and Eugene Charniak. 2000. Assigning Function Tags to Parsed Text. *Proceedings of the North American Chapter of the Association of Computational Linguistics.*

Eugene Charniak. 2000. A Maximum-entropy-inspired Parser. *Proceedings of the North American Chapter of the Association of Computational Linguistics*.

Michael Collins. 2000. Discriminative Reranking for Natural Language Parsing. Machine Learning: *Proceedings of the Seventeenth International Conference (ICML 2000),* Stanford California. USA

Melanie Gnasa and Jens Woch. 2002. Architecture of a knowledge based interactive Information Retrieval System. http://konvens2002.dfki.de/cd/pdf/12P-gnasa.pdf

Noriko Kando. 2000. What Shall we Evaluate? Preliminary Discussion for the NTCIR Patent IR Challenge (PIC) Based on the Brainstorming with the Specialized Intermediaries in Patent Searching and Patent Attorneys. *Proceedings of the ACM SIGIR 2000 Workshop on Patent Retrieval* in conjunction with *The 23rd Annual International ACM SIGIR Conference on Research and Development in Information retrieval*. Athens. Greece

Svetlana Sheremetyeva and Sergei Nirenburg. 1996. Generating Patent Claims. *Proceedings of the 8th International Workshop on Natural Language Generation*. Herstmonceux, Sussex, UK.

Svetlana Sheremetyeva 2003. Towards Designing Natural Language Interfaces. *Proceedings of the 4th International Conference "Computational Linguistics and Intelligent Text Processing"* Mexico City, Mexico