Handling Low Translatability in Machine Translation of Long Sentences

Svetlana SHEREMETYEVA

LanA Consulting ApS
Mynstersvej 7A, 2
Copenhagen, Denmark, DK-1827
lanaconsult@mail.dk

Abstract. Long and complex sentences are normally "low translatable" sentences due to high ambiguity. We describe a methodology of handling such sentences in APTrans, - a system for machine translation of patent claims between English and other languages. APTrans builds on patent data hard-coded in the system lexicon and grammars coached in diverse grammatical frameworks (PSG and predicate/argument DG). We motivate a specific partial parse, - in our analyzer parsing is reduced to a phrase level and a level of individual simple clauses. The load of detecting a clause hierarchy in a complex sentence is shifted to the generator. Transfer combines an interlingual and a syntactic transfer approaches. The methodology is universal in the sense that it could be used for different domains, languages and applications. We introduce special features of APTrans on the example of English/ Danish language pair.

1. Introduction

Low translatability indicators, such as - included and parallel structures, ambiguous PP attachments, etc., (Underwood and Jongejan, 2001) characteristic of long sentences, are the reasons that currently no commercial MT system can translate patent claims adequately.

A device for producing a spray of electrically charged particles ***comprising***
means ***defining*** *a location from which the spray* ***is generated****, and*
a voltage generator for producing high voltage between said location and the surroundings, characterized in that said voltage generator ***comprises*** *a large solid state array of radiation sensitive voltage* ***producing*** *elements* ***interconnected*** *to produce high voltage...*

Fig. 1. A Fragment of a US patent claim. Predicates of included clauses are bold-faced.

In many MT systems long sentences are broken up along punctuation, and the segments are parsed separately (Kim and Ehara, 1994). This method can be incorrect due to the punctuation ambiguity. The long sentence problem is sometimes approached by being very selective about which sentences to parse as in (Hobbs and Bear, 1995), where statistical filter is used to pre-process a text. In the Pattern-based English-Korean MT (Roh et al., 2003) long sentences are handled by using chunking information on the phrase-level of the parse result to which a sentence pattern is applied directly. A patent specific research in MT where the problem of low translatable sentences is addressed by suggesting an interactive analysis module has been done for Russian to English by (Sheremetyeva and Nirenburg, 1999). The most recent

attempt to cope with low translatability of a patent claim is a Japanese-English patent MT system, which merges the English claim authoring system AutoPat (Sheremetyeva, 2003) and a Japanese PC-Transfer application (Neumann, 2005).

The APTrans system presented in the current paper integrates some of the transfer and generation techniques described in the last cited works but relies on an automatic analyzer bypassing some of low translatability problems. 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 (NP, PP, etc.) level does not usually generate more ambiguity as a sentence becomes longer (Abney, 1996). The specificity of our approach is that parsing is not required to produce the structural information of higher levels than a simple clause in the syntactic tree of a complex claim structure. The parser carries out the analysis on a phrase level and a level of individual simple clauses, which results in an interlingual content representation. The load of detecting a clause hierarchy is shifted to the generator. The system is augmented with domain tuned proofing tools: spelling and grammar checkers. APTrans draws heavily on patent data that, as our research showed, feature great similarity across many languages, i.e., sublanguages of different national languages in patent domain are much closer than these languages as such. The linguistic knowledge of the system currently covers English and Danish, but the methodology, engine programs and developer tools make APTrans easily extendable to any other language pair.

2. Lexicon

APTrans lexicon contains rather deep knowledge crucial for all components of APTrans. It includes corpus-based cross-referenced monolingual lexicons. Every monolingual lexicon consists of a set of single sense entries maximally defined as a tree of the following features:

Every entry is maximally defined as a tree of features:

SEM-CL[Language[POS RANK [MORPH CASE_ROLE FILLER PATTERN]

SEM_Cl - semantic class;

CASE_ROLEs, - a set of case roles associated with a lexeme, if any;

FILLERs - sets of most probable fillers of case-roles in terms of types of phrases and lexical preferences (field "case-role syntax" in Figure 2).

PATTERNs - patterns that code both the knowledge about co-occurrences of lexemes with their case-roles and the knowledge about their linear order (local word order) (*linking feature*s), e.g., the pattern (13 x 2 4) for the predicate *"connected"* (see Figure 2) means that this predicate can have case-roles 1(subject), 2(indirect-object), 3(manner) and 4(purpose) realized simultaneously and in such a case the order of the words should be: "*1: wires 3:electrically x: connected 2:to the lamp 4: to switch it off"*

POS - part of speech out of the set of 14 POS defined for the domain. To simplify processing we consider passive and active predicates as belonging to different parts of speech.

MORPH - explicitly listed domain relevant wordforms, number, gender, etc., (*morphological feature*s). The beauty of the claim domain is that verb (predicate) paradigms are very much restricted and we can save acquisition effort on listing only those wordforms, which can occur in the claim text and skip those which do not (see more on claim sublanguage analysis in and its representation in TransDict in (Sheremetyeva, 2005).

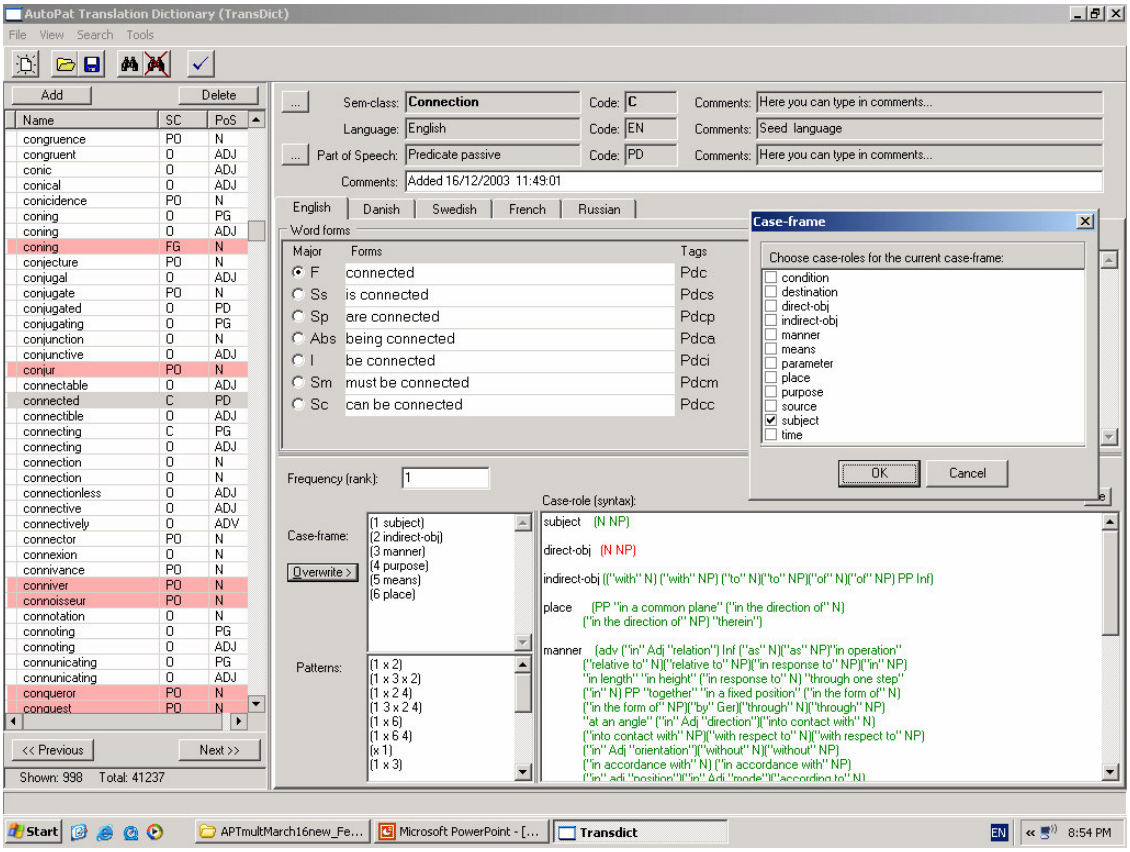

Figure 2. A screenshot of the developer TransDict interface with a typical maximal entry for the predicate. Clicking on language bookmarks over the "Word forms" field will open entries equivalent to "connected" in different languages. Every word form is associated with a supertag (shown on the right of the word form field), which will be assigned to the word during text tagging.

Figure 2 shows a self-explanatory screenshot of the TransDict developer interface with the (maximal) entry for the predicate "connected". All seven domain relevant wordforms of the lexeme is associated with a specific supertag[1] coding deep linguistic knowledge.

For example, the supertag "Pdcs" of the wordform "is connected" in Figure 2 means that this wordform is a verb from the semantic class "connection" in a finite form, present, singular, passive voice. In general, a supertag in our system is a typed feature structure; the set of features assigned by every supertag is application and domain dependent. This allows us both to provide for the adequate disambiguation power of the analyzer, and to avoid a situation when too fine grain size of features in tags as well as a large number of tags may lead to computational problems (see e.g., Church, 1988). We currently 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.

All monolingual entries are cross-referenced with equivalent entries in other languages.

[1] Joshi and Srinivas (1994) who seem to coin the term « supertag » use elementary trees of Lexicalized Tree-Adjoining Grammar for supertagging lexical items. We use the term « supertag » in a different meaning to just indicate that it codes the knowledge richer than a POS, namely a set of features defined in our lexicon.

3. Grammar

The grammar in our system is a mixture of context free strongly lexicalized Phrase Structure Grammar (PG) and Dependency Grammar (DG) formalisms. The PG component covers only those linguistic entities that are *neither predicates, nor clauses* in a complex sentence. The second component of our grammar is a Case-Role Dependency Grammar (Fillmore, 1970). 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 and predicates as specified in the lexicon.

The grammars assign clauses a representation as shown in Figure 3, 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 lexicon. *Word* and *phrase* are a word and phrase (NPs, PPs, etc.) as specified by PG grammar.

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

Figure 3. An interlingual representation of a claim sentence.

The APTrans parser, as mentioned above, is not required to produce a full parse of a complex claim sentence, which could have been too ambiguous. The parser assigns syntactic structures to clause constituent phrases and skipping a complete syntactic parse of a clause represents the clause structure in terms of predicate/argument dependencies. The output of the parser is a set of interlingual predicate/argument structures representing separate claim clauses with no information about their hierarchy in the claim sentence.

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.

The parsing module includes a tokenizer, a supertagger, a bottom-up heuristic parser, and a deep semantico-syntactic parser.

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 are used to augment the resolution power of disambiguation rules.

Supertagging generates all possible supertags for a word. As it is crucial for our system to have multiple supertags correctly disambiguated supertagger includes a powerful disambiguation module with constraint-based domain specific rules. Rules discarding faulty readings of ambiguously tagged words rely on the knowledge in the lexicon, lexical and tag preferences, 5 step context of "supertags" and "boundary" tags.

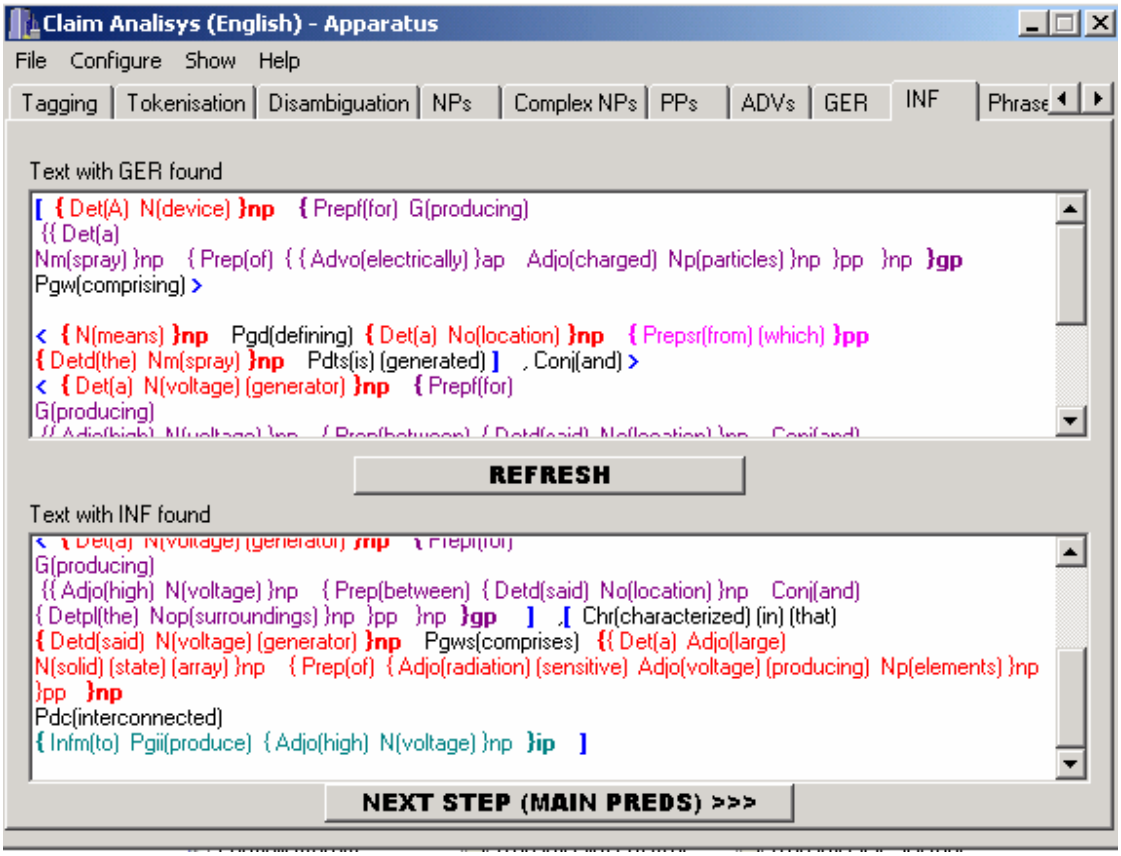

Figure 4. **A screenshot of the developer interface showing the traces of bottom-up heuristic parsing based on PG grammar formalism**.

Bottom-up heuristic parsing is a recursive pattern matching of supertag strings against the right hand sides of the rules in the PG component of our grammar. This 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 parser can disambiguate such complex NPs as "coupling locations of reference and support means". After complex NPs boundaries are placed other types of phrases (PPs, AdvPs, GerPs, InfPs) are identified in turn.

For example, the rule used to identify the Gerundial phrase:

{ Prepf(for) G(producing) {{ Det(a) Nm(spray) }np { Prep(of) { { Advo(electrically) }ap Adjo(charged) Np(particles) }np }pp }np }gp

looks like:

```
IF T =
~PR{1;ListGer}~Ger{1}~Bnp{1;ListOpen}
~ANY{+}~Bnp{1;ListClose}
THEN
   BRACKETS
"Bgp0(open)","Bgp0(close)"
```

This rule reads, "If a phrase starts with a preposition from a list of prepositions specified for Gerunds ("for", "by" in our specification) or one Gerund followed by one NP opening border tag followed by one or more supertags specified in the set ANY, which, in turn, is followed by the NP closing border tag, then put the Gerund phrase opening border tag at the beginning

of this phrase and the Gerund closing border tag at the end".

The set of rules are ordered based on a set of heuristics. As can be seen from examples, our parser does not only identify coherent word sequences as most typical chunkers but also detects the internal structure of chunks. A screenshot of the developer interface showing the traces of bottom-up heuristic parsing based on PG grammar formalism is given in Figure 3.

Identifying predicates 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. At this step in addition to PG we start using our DG mechanism and predicate/argument knowledge stored in the lexicon.

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.

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 the 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.

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 problems, which to a great extend can be corrected with data driven heuristics, e.g., the probabilistic knowledge about case-role weights from the lexicon given the meaning of a predicate.

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 claim sentence against those present in all homonymous predicate entries. A special metrics is developed to make disambiguation decisions.

Correct case-role procedure attempts to correct a case-role status of a phrase if it does not fit the predicate description in the lexicon.

A fragment of the analyzer parse for our example (see Figure 1) is shown in the left pane of Figure 5.

4. Transfer

The input to the transfer module is a set of SL (English in our case) predicate/argument structures with syntactically parsed case-role fillers as shown in the left pane in Figure 4.

The APTrans transfer procedure is based on a combination of interlingual and syntactic transfer approaches and is in fact reduced to the translation of phrases, - case-role fillers. The first step is to substitute every SL predicate with its TL equivalent from the lexicon, the set of SL case-roles is considered to be semantic invariant with respect to transfer. The results of this step (called "base transfer" in shown in the right pane of Figure 4). The second transfer step,

translation of case-role fillers, is done by means rule-based syntactic transfer. We bypass the problems of morphological generation by retrieving a rule specified wordform from the morphological field of the lexicon where all the relevant wordforms for a lexeme are listed explicitly. The output of the transfer module is a set of TL (Danish) predicate templates with fully translated Danish case-role fillers as shown in the right pane of Figure 5 and in the left pane of Figure 6.

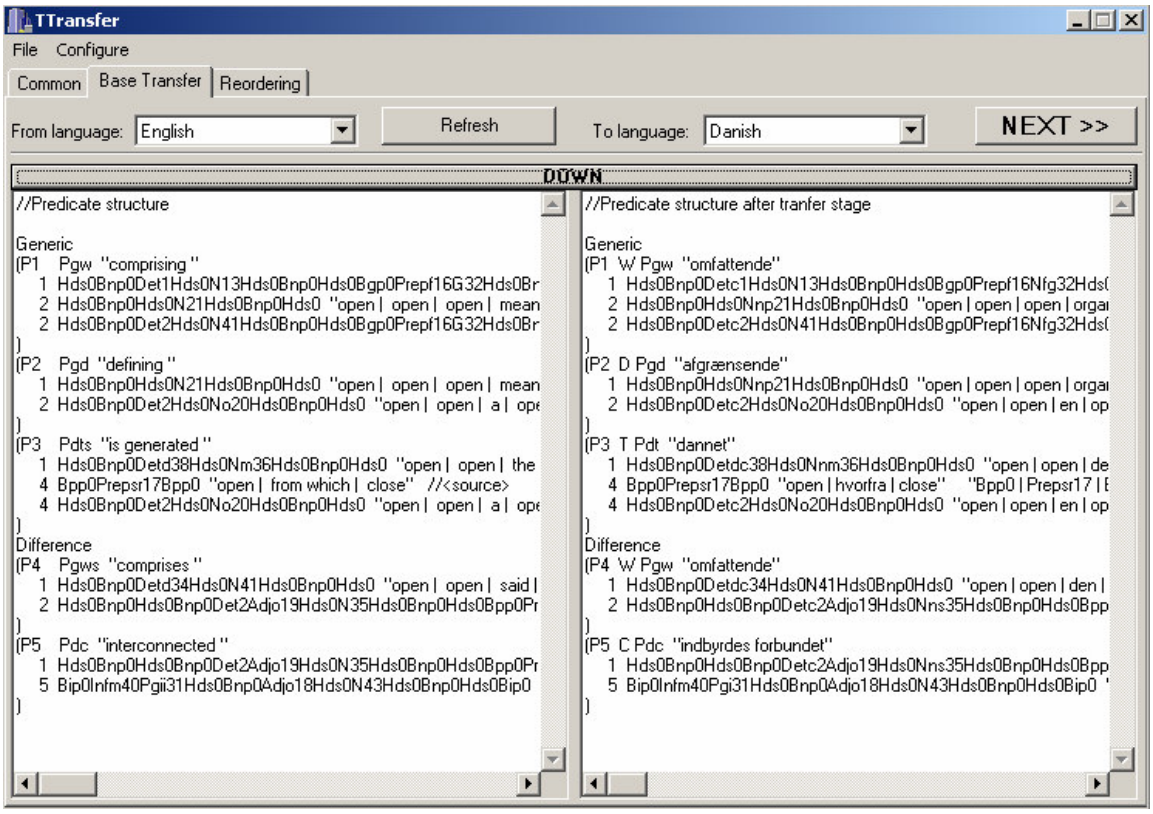

Figure 5. **A screenshot of the developer interface showing traces of the English parser output (on the left) and the results of the first transfer stage (on the right). Predicate structures are invariant, English lexemes are substituted by the base forms of equivalent Danish lexemes from the lexicon. Danish strings filling predicate case-roles are then matched against syntactic transfer patterns to provide for complete translation of case-roles as shown in the left pane of Figure 5.**

5. Generation

APTrans generator takes the output of the transfer module as input and produces a TL (currently Danish) translation of an English claim, as illustrated in Figure 6.

APTrans generation module to a large extent reuses fully operational English generator from a different patent related application, AutoPat, a computer system for authoring patent claims (Sheremetyeva, 2003).

Generation process consists of several stages with different inner representations of a patent claim at every stage, each generation stage fulfilling a special task. Like the parse the generation is pursued "best first" decision according to a set of *heuristics* compiled through lots of experience generation.

The generator first creates an hierarchical structure of TL predicate templates in the form of a root tree or a forest of root trees out of individual templates. This is done by clustering the

templates describing the same elements of invention (case-roles) and ordering them according to the three weighted parameters: the order of predicates in the SL text, rhetorical requirements and stylistic requirements to the claim text. The tree building algorithm is hard-coded into the program and is language-independent. It to a large extent relies on the legal knowledge about patent claims coded in the lexicon. At the second stage of generation the forest of predicate trees is linearised by top-bottom depth-first bypassing algorithm, which results in a bracketed string of characters (see left bottom pane in Figure 6).

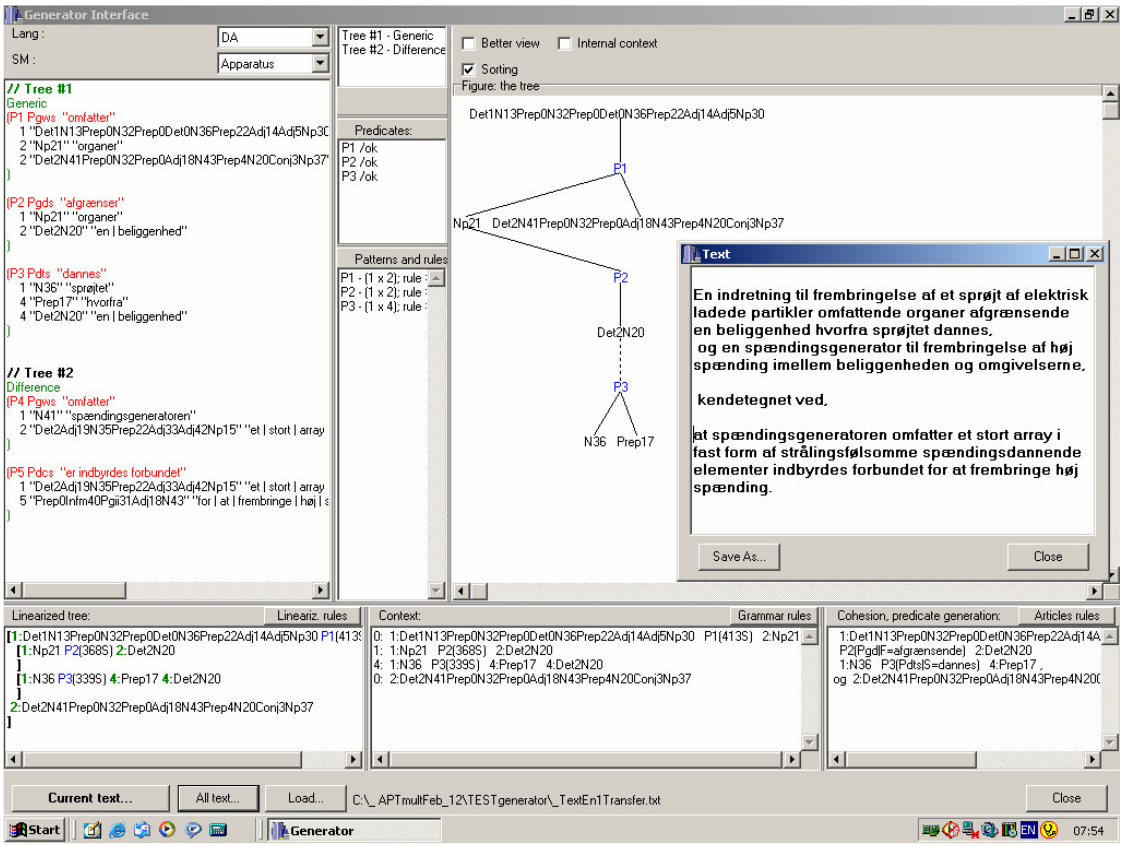

Figure 6. A screenshot of the developer generator interface showing traces of the generation procedure. The output of the transfer stage (a set of predicate structures with fully translated and tagged Danish case-role fillers) is shown in the left pane. A tree of the "glued" input structures built by the generator to specify the hierarchy of the claim clauses is displayed in the right background pane. Three bottom windows show the traces of successive stages of the generation procedure as described in (Sheremetyeva, cf). The resulting Danish translation of the English claim in Figure 1 is given in the pop-up window on the right.

The brackets specify the hierarchy of predicate templates (i.e., constituent claim clauses), thus making up for the incomplete parse of the analyzer. At this linearization stage two decisions are made, - the first specifies the linear order of a predicate and its case-roles for each predicate template, and the second determines the location of a newly linearised predicate template in the already existing string of predicate structures. For example, of one of the rules making the first decision can be:

IF NOT (CP=PP)
THEN
  CP=MOSTFREQ

The rule reads, “If the predicate of the currently bypassed template in the tree is not a preposition then linearise the current template following the linear pattern containing corresponding case-roles from the predicate entry of the lexicon”.

Linear patterns selected for the predicate templates in our example are given in the middle pane in Figure 6.

A new segment can be either inserted into the existing linearised structure at a certain point or simply concatenated to it at the end. For example, one of the linearization rules responsible for the second decision is:

```
IF CLASS="A"
THEN
   IF  ISLEFTMOST
   THEN
      INSERT NEXTCR
   ELSE
      INSERT NEXTPRED
```

This rule reads as follows: “If the predicate of the current (bypassed) template in the tree of templates belongs to the semantic class “ meronymy” (*comprising, having, including, etc.*) and if this predicate template is leftmost in the bush of siblings then insert a linearised segment of the current template into the linearised segment of the parent predicate template next to the case-role by which the current template was linked to the parent template. If the current predicate template is not left most then insert its linearised segment into the existing string next to the linearised segment of the parent predicate”.

Linearization is based a set of rules that strictly speaking should be language-dependent but in practice (due to interlingual sublanguage similarity) are almost universal.

At the third generation stage, - realization, the linearised bracketed string of characters is passed from left to right, and based on a special set of heuristic rules procedures taking care of ellipsis, conjoint structures, punctuation and morphological forms of predicates are executed. The result is a claim text in a TL meeting all legal requirements.

6. Conclusions and future work

We presented a grammar-based data-intensive MT system, APTrans, capable of handling low translatable sentences, such as patent claims. The specificity of our approach is that parsing is not required to produce the structural information of higher levels than a simple clause in the syntactic tree of a complex claim structure. The parser carries out the analysis on a phrase level and a level of individual simple clauses, which results in an interlingual content representation. The load of detecting a clause hierarchy is shifted to the generator. The system is augmented with domain tuned proofing tools: spelling and grammar checkers.

APTrans is now implemented in its demo version for the English/Danish language pair. Due to high cross-language similarity of patent claims and the design of our software we could simply update English generation rules from a different patent related application, AutoPat (a computer system for authoring patent claims) for the target Danish language. Most of the rules were reused, thus saving a lot of development effort and time.

We have not yet made a large-scale evaluation of our system. This leaves the comparison between other MT systems and our APTrans as a future work. Preliminary results show a reasonably small number of failures, mainly due to the incompleteness of rules and lexicon. We are currently concentrated on increasing the coverage of the system and intend to include more languages.

In general, our methodology and experience in developing patent related applications for Russian, English, Danish and Japanese (Neumann, 2005) lets us believe that covering every new pair of languages by APTrans will take much less development effort and time than the first pair of languages.

7. References

ABNEY, S (1996). Part-of-speech Tagging and Partial Parsing. In: Corpus-Based methods in Language and Speech. Klue Academic Publishes

FILLMORE Charles J. (1970). Subjects, speakers and roles. Synthese.21/3/4

CHURCH K.W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. *Second Conference on applied Natural Language Processing, Austin, Texas.*

HOBBS J. R., and BEAR J. (1995). Two Principles of Parse Preference. In: Linguistica Computazionale Current Issues in Computational Linguistics: In Honour of Don Walker, Vol.9

JOSHI A., and SRINIVAS B. (1994). Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing.

http://acl.ldc.upenn.edu/C/C94/C94-1024.pdf.

KIM Y., EHARA T. (1994). A Method for Partitioning of Long Japanese Sentences with Subject Resolution in J/E Machine Translation. In: Proceedings of the 1994 ICCPOL

NEUMANN Ch.(2005). A Human-Aided Machine Translation System for Japanese-English Patent Translation. Proceedings of the Workshop on Patent Translation in Conjunction with MT Summit, Phuket, Thailand, September 16.

ROH Y., HONG M., CHOI S., LEE K., PARK S. (2003). For the Proper Treatment of Long Sentences in a Sentence Pattern-based English-Korean MT System. In; Proceedings of MT Summit IX. New Orleans

SHEREMETYEVA, S. and S. NIRENBURG. (1999). Interactive MT As Support For Non-Native Language Authoring. Proceedings of the MT Summit VII. September 13-17, 1999, Singapore.

SHEREMETYEVA S. (2003). Towards Designing Natural Language Interfaces. Proceedings of the 4th International Conference "Computational Linguistics and Intelligent Text Processing" Mexico City, Mexico, February 16-22.

SHEREMETYEVA S. (2005). "Less, Easier and Quicker" in Language Acquisition for Patent MT. *Proceedings of the Workshop on Patent Translation in conjunction with the MT Summit X, September.* Phluket, Thailand

UNDERWOOD N. L and JONGEJAN B. (2001). Translatability Checker: A Tool to Help Decide Whether to Use MT. Proceedings of MT Summit VIII, Santiago de Compostela, Spain.