UPPSALA
UNIVERSITET

Integrating a dependency parser into a system for machine translation

Lina Stadell

Uppsala University
Department of Linguistics and Philology
Språkteknologiprogrammet
(Language Technology Programme)
Bachelor's Thesis in Language Technology

13th December 2011

Supervisors:
Joakim Nivre, Uppsala University
Eva Pettersson, Convertus AB
Anna Sågvall Hein, Convertus AB

Abstract

This thesis investigates the possibilities of transforming dependency structures into feature value structures, in order to replace a unification-based parser with a dependency parser in a machine translation system.

A transformation component that uses hand-crafted rules to map between structures was developed. This component, together with MaltParser, was used to replace the unification-based parser in the translation system. The resulting system was tested on course syllabi data and compared with the original system.

The test results showed that about a third of the test sentences was succesfully transformed. Translation quality for these sentences was worse than for the original system, but still acceptable. The results indicate that the integration of a dependency parser into a machine translation system operating on feature value structures is possible, although the transformation component needs to be improved further before being put to use.

Contents

Acknowledgments

I would like to thank my supervisors, Joakim Nivre at Uppsala University and Eva Pettersson and Anna Sågvall Hein at Convertus AB, for their time, support and good advice. I would also like to thank Per Weijnitz at Convertus AB for technical support. Finally, a big thank you to Niklas Malmgren for proof reading, comments and encouragement.

1 Introduction

1.1 Purpose

The purpose of this thesis is to examine the possibilities of transforming dependency structures into feature value structures, in order to replace a unification-based parser with a dependency parser in a transfer-based system for machine translation. The system in question is in use by the language technology company Convertus, and currently uses Uppsala Chart Parser for syntactic analysis. In order to expand its services into translations from other languages than Swedish, Convertus wishes to look into the possibilities of replacing the parser component in the system, while keeping the other components as they are. I will attempt to integrate MaltParser into the system, and evaluate the results with regard to translation quality.

The integration will be done by implementing a transformation module capable of converting dependency structures into equivalent feature structures. This module will, together with MaltParser, replace the current parser module in the translation system. If successful, this approach will keep modifications of the current system to a minimum. The transformation from dependency structures to feature value structures means that it is possible to keep the transfer and generation rules in the system as they are, with perhaps a few minor modifications.

The transformation module will be closely tied to a specific language and annotation scheme, i.e. the one used in the training data for MaltParser. I will however attempt to make the implementation as generic as possible, allowing for the possibility of adapting the module to deal with other languages at a later point. For the purpose of this thesis the implementation will be done for translations from Swedish into English, which makes it possible to compare the results against the existing Convertus system.

1.2 Outline of this Thesis

Chapter 2 gives a background to the software used for this investigation and the data used for training and parsing. It also gives a brief overview of dependency parsing and unification-based parsing. In chapter 3 I describe the implementation of the transformation program and discuss some issues that arose during the implementation process. Chapter 4 describes the test setup and presents the results. Chapter 5 contains a discussion of the results while conclusions can be found in chapter 6.

2 Background

The field of machine translation can be divided into two major areas, rule-based machine translation and statistical machine translation.

In rule-based machine translation the transfer from source to target language is done by consulting linguistic resources such as dictionaries and grammars. The translation includes three steps: first a syntactic analysis is made of the source language sentence, which is then transferred into a syntactic representation in the target language by applying a set of transfer rules. Finally, the target sentence is generated by applying a set of generation rules.

Statistical machine translation is based on the idea that the goal of translation is to produce an output that maximizes some value function representing the quality of the translation (Jurafsky and Martin, 2009). Translation is done by building probabilistic language models which are used to choose the most probable translation for an input sentence.

Syntactic parsing plays an important role in rule-based machine translation, which relies on a syntactic analysis of the source language in order to produce translations.

2.1 Convertus Syllabus Translator

The machine translation system used by Convertus is a rule-based transfer system with statistical fall-back approaches. It is based on the MATS system, which was developed in a research project at the Department of Linguistics at Uppsala University (Sågvall Hein et al., 2003). The MATS system started out as a pure rule-based system, but was later enhanced with statistical fall-back methods to increase robustness. (Weijnitz et al., 2004)

Convertus started as a spin-off from the research group working on the MATS system. The company has continued to develop the system as a commercial product, the Convertus Syllabus Translator, which is a machine translation service specialized in the translation of course syllabi and currently in use by a number of Swedish universities. The system implementation is modular, with separate components responsible for pre-processing, analysis, transfer and generation. An overview of the processing steps in the system can be seen in figure 2.1. The current system translates from Swedish to English, but there are hopes that the integration of MaltParser will enable translations from other source languages.

The core of Convertus' translation system is the MULTRA engine, which is responsible for transfer and generation. MULTRA is a transfer-based translation engine that works with unification of feature structures (Sågvall Hein, 1997).

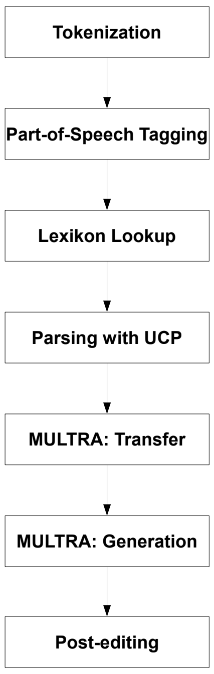

Figure 2.1: Processing steps in Convertus' system

The rules for transfer and generation are expressed in a formalism developed by Beskow (1993), as an extension of the PATR-II developed by Shieber (2003). The rules in MULTRA are context-free, where the left-hand side consists of a feature structure and the right-hand side of zero or more feature structures. The relation between the feature structures are described by a series of identity equations.

2.2 Unification-based Parsing

Unification-based parsing constructs a syntactic analysis by the unification of feature structures. Shieber (2003) defines a feature structure as a partial function from features to their values. An example could be the function mapping the feature *number* onto the value *singular*. This feature structure could itself be a value contained in a larger structure. *Unification* combines the information from two feature structures, forming a new structure that contains all the information of the unified structures but no additional information. If the two feature structures contain conflicting information, unification fails.

Feature structures can be thought of as directed acyclic graphs (DAGs), where the arcs are labeled with feature names and points to nodes that can be either atomic values or another feature structure. The unification of two DAGs can be done by searching through the graph representation of two input segments, trying to find features that match and altering the graphs so that the feature arcs from both segments point to the same value node (Jurafsky and Martin, 2009).

Parsing based on unification can be done by means of a context-free grammar, augmented with unification constraints. The algorithm used for searching can be any standard parsing algorithm.

2.2.1 Uppsala Chart Parser

Syntactic analysis in MULTRA is performed by a unification-based parser component, the Uppsala Chart Parser (UCP). It is a procedural chart parser operating on a set of hand-crafted grammar rules. The Uppsala Chart Parser was originally coded in LISP (Sågvall Hein, 1982) but was later re-coded by Weijnitz (2002), with the objective of speeding up parsing. The output from UCP consists of feature structures that are passed on to MULTRA for transfer and generation. Figure 2.2 shows an example of a feature structure analysis for the sentence *De etiska frågeställningarna kommer att belysas.*.

2.3 Dependency Parsing

Dependency parsing is based on the idea that the syntactic structure of a sentence can be described by analysing the relations between words. These relations are thought to hold between two words, the *head* and the *dependent*. The dependency relations can be unlabeled or labeled with categories describing the syntactic function of the dependent.

For the purpose of dependency parsing we consider this syntactic representation a dependency graph – a labeled, directed graph where the nodes correspond to the tokens of the sentence being parsed and the arcs constitute the dependency relations between tokens. With a formal definition taken from Nivre (2006): given a set R of dependency types, a *dependency graph* for a sentence $x = \{w_1, ..., w_n\}$ is a labeled directed graph $G = (V, E, L)$ where:

1. $V = Z_{n+1}$
2. $E \subseteq V \times V$
3. $L : E \to R$

The set of nodes V consists of indexes for the tokens in the sentence with the special root node bearing index 0. This node is a technical construct without a corresponding token in the input sentence. The set of arcs E consists of ordered pairs of nodes (i, j), where i is the *head* and j is the *dependent*. L is a function that assigns a dependency type $r \in R$ to every arc $e \in E$ in the dependency graph.

The dependency tree should be connected and acyclic with no more than one head for every node. For the work done in this thesis the tree should also be projective, which means that for every arc $i \to j$ there should also be a path from i to k for every node k such that $i < k < j$ or $j < k < i$. Or, put more simply, there can not be any crossing arcs in the graph.

```
[ phr.cat : sent
  1 : [ phr.cat : cl
        cl.type : main
        mode : decl
        verb : [ diat : act
                 inff : fin
                 verb.type : main
                 tense : pres
                 head : [ lex : [ sym : komma.vb.1
                                  val : va.komma ]
                          trglex : [ sym : come.vb.1 ]
                          pv : −
                          word.cat : VERB ] ]
        subj : [ phr.cat : np
                 def : def
                 form : def
                 numb : plur
                 det : [ lex : [ sym : den.al.1 ]
                         trglex : [ sym : the.al.1 ]
                         word.cat : ART
                         det.type : det
                         def : def
                         numb : plur ]
                 gender : utr
                 sex : n
                 case : basic
                 person : 3
                 df : [ phr.cat : ng
                        numb : plur
                        gender : utr
                        attr : [ phr.cat : adjp
                                 head : [ word.cat : ADJ
                                          degree : pos
                                          gender : nil
                                          lex : [ sym : etisk.av.1 ]
                                          trglex : [ sym : ethical.av.1 ] ]
                                 numb : plur
                                 form : def
                                 case : basic
                                 gender : utr ]
                        sex : n
                        form : def
                        case : basic
                        person : 3
                        head : [ lex : [ sym : frågeställning.nn.1 ]
                                 trglex : [ sym : issue.nn.1 ]
                                 word.cat : NOUN ]
                        np.type : head.lex ] ]
        obj.inf : [ phr.cat : cl
                    cl.type : inf
                    inf.mark : [ word.cat : IE
                                 lex : [ sym : att.ie.1 ]
                                 trglex : [ sym : to.ie.1 ] ]
                    verb : [ diat : pass
                             inff : inf
                             verb.type : main
                             head : [ lex : [ sym : belysa.vb.1
                                              val : va.plundra ]
                                      trglex : [ sym : enlighten.vb.1 ]
                                      pv : −
                                      word.cat : VERB ] ] ] ]
  sep : [ word.cat : SEP
          type : major
          lex : [ sym : stop.sr.1 ]
          trglex : [ sym : stop.sr.1 ] ] ]
```

Figure 2.2: Feature structure analysis for the sentence *De etiska frågeställningarna kommer att belysas.*

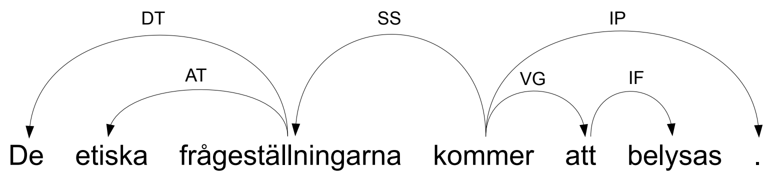

Figure 2.3: Dependency structure analysis for the sentence *Det etiska frågeställningarna kommer att belysas.*

Given this definition of dependency graphs, *dependency parsing* can be defined as the mapping between text sentences and dependency graphs.

2.3.1 MaltParser

MaltParser is a system for dependency parsing developed at Växjö University and Uppsala University (Nivre et al., 2007). It uses treebank data to induce a parsing model and parses new data using the induced model. The system is modular with respect to parsing algorithms, feature models and learning methods and gives the user the flexibility to freely choose which components are best suited for the application at hand. Figure 2.3 shows a dependency structure analysis for the same example sentence as in figure 2.2.

2.4 Swedish TreeBank

The MaltParser setup in this thesis uses the pre-trained Swedish model available from `www.maltparser.org` for parsing. The data is from the Swedish Treebank, a joint corpus consisting of Talbanken (Einarsson, 1976a) (Einarsson, 1976b) and the Stockholm-Umeå Corpus (SUC, 1997). The model is trained on the Talbanken part of the corpus, which has been supplied with dependency annotation by Nivre et al. (2006).

3 Implementation

The integration of MaltParser into Convertus' translation system is dependent on a transformation component that is capable of transforming the output from MaltParser into structures that can be handled by the MULTRA transfer and generation rules. An attempt will be made to construct such a component, which together with MaltParser could replace the UCP parser module in Convertus' system.

This chapter gives a step-by-step walkthrough of the translation of an example sentence by the MaltParser-based system, as well as a detailed description of the implementation of the transformation component. Figure 3.1 gives an overview of the processing steps in the MaltParser-based system. The first steps, tokenization, part-of-speech tagging and dictionary lookup, are technically the same as in Convertus' system, although the implementation of them differs

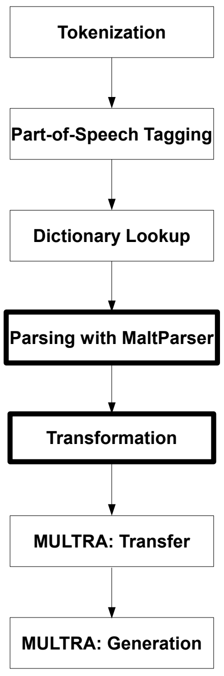

Figure 3.1: Processing steps

slightly. The bold boxes show the parsing and transformation steps, which replaces parsing with UCP in figure 2.1. Transfer and generation in MULTRA is the same. The post-editing module was not used in either of the systems for the purpose of this thesis.

3.1 Tokenization

For the purpose of this thesis I have written a very simple tokenizer, which produces the output seen in figure 3.2. For future purposes this tokenizer should be replaced with Convertus' own tokenizer, which is adapted to the domain of course syllabi.

```
De
etiska
frågeställningarna
kommer
att
belysas
.
```

Figure 3.2: Output after tokenization

3.2 Part-of-Speech Tagging

Part-of-Speech tagging is done using the open source HMM-tagger HunPos (Halácsy et al., 2007) with the default language model for Swedish. Figure 3.3 shows output for the example sentence after tagging.

```
De                     DT|UTR/NEU|PLU|DEF
etiska                 JJ|POS|UTR/NEU|PLU|IND/DEF|NOM
frågeställningarna     NN|UTR|PLU|DEF|NOM
kommer                 VB|PRS|AKT
att                    IE
belysas                VB|INF|SFO
.                      MAD
```

Figure 3.3: Output after tagging

3.3 Parsing

The next step in the translation process is parsing with MaltParser. The output is in the CoNLL data format, as in figure 3.4. Each line consists of tab-separated fields that hold information about the token position in the sentence, its word form, part-of-speech tag and morphological features, its head word in the dependency tree and the type of dependency relation to the head.

```
1    De                  DT    TR/NEU|PLU|DEF               3    DT
2    etiska              JJ    POS|UTR/NEU|PLU|IND/DEF|NOM 3    AT
3    frågeställningarna NN    UTR|PLU|DEF|NOM              4    SS
4    kommer              VB    PRS|AKT                      0    ROOT
5    att                 IE    -                            4    VG
6    belysas             VB    INF|SFO                      5    IF
7    .                   MAD   -                            4    IP
```

Figure 3.4: Output after parsing

3.4 Dictionary Lookup

The final step before transformation is a dictionary lookup, where the input text is sent to the Convertus Translator lexicon module for extraction of lexical information. This lookup produces a dictionary file (see figure 3.5) containing all matching dictionary entries for each word in input, which is later used in the transformation process.

3.5 Transformation

The transformation component takes the CoNLL-output from MaltParser and transforms it into an XML file containing feature structures, which can be handled by the transfer and generation modules in MULTRA. This involves assigning part-of-speech tags and morphological features a feature-value representation, inserting phrase nodes and constructing a syntactical representation that is compatible with MULTRA. The transformation component is also responsible for incorporating the target language lexical information that can be found in the dictionary file. Figure 3.6 shows the output for the example sentence.

The transformation component deals with one token at a time. Transformation is done in two steps. First, tokens are converted into a feature-value representation by looking them up in mapfiles consisting of rules mapping between source and target structures. At the same time information about the phrase structure for the token is added. Secondly, syntax is built by concatenating tokens into a feature structure analysis of the whole sentence.

```
<s id="0">
        <lexdat index="0" token="De" id="0" db="general" length="1" mosy="ALXPD" >
                <lem>
                        <sym>den.al</sym>
                        <lex><sym>den.al.1</sym></lex>
                        <trglex><sym>the.al.1_id0</sym></trglex>
                </lem>
        </lexdat>
        <lexdat index="0" token="De" id="1" db="general" length="1"mosy="PNPXPS3">
                <lem>
                        <sym>de.pn</sym>
                        <lex><sym>de.pn.1</sym></lex>
                        <trglex><sym>they.pn.1_id1</sym></trglex>
                </lem>
        </lexdat>
        <lexdat index="1" token="etiska" id="2" db="hsv" length="1" mosy="AVXPXBP" >
                <lem>
                        <sym>etisk.av</sym>
                        <lex><sym>etisk.av.1</sym></lex>
                        <trglex><sym>ethical.av.1_id2</sym></trglex>
                </lem>
        </lexdat>
        <lexdat index="1" token="etiska" id="3" db="hsv" length="1" mosy="AVXSDBP" >
                <lem>
                        <sym>etisk.av</sym>
                        <lex><sym>etisk.av.1</sym></lex>
                        <trglex><sym>ethical.av.1_id3</sym></trglex>
                </lem>
        </lexdat>
        <lexdat index="2" token="frågeställningarna" id="4" db="general" length="1" mosy="NNUPDB" >
                <lem>
                        <sym>frågeställning.nn</sym>
                        <lex><sym>frågeställning.nn.1</sym></lex>
                        <trglex><sym>issue.nn.1_id4</sym></trglex>
                </lem>
        </lexdat>
        <lexdat index="3" token="kommer" id="5" db="general" length="1" mosy="VBAPM" >
                <lem>
                        <sym>komma.vb</sym>
                        <lex><sym>komma.vb.1</sym><val>va.komma</val></lex>
                        <trglex><sym>come.vb.1_id5</sym></trglex>
                </lem>
        </lexdat>
        <lexdat index="4" token="att" id="6" db="general" length="1" mosy="IE" >
                <lem>
                        <sym>att.ie</sym>
                        <lex><sym>att.ie.1</sym></lex>
                        <trglex><sym>to.ie.1_id6</sym></trglex>
                </lem>
        </lexdat>
        <lexdat index="4" token="att" id="7" db="general" length="1" mosy="SN" >
                <lem>
                        <sym>att.sn</sym>
                        <lex><sym>att.sn.1</sym></lex>
                        <trglex><sym>that.sn.1_id7</sym></trglex>
                </lem>
        </lexdat>
        <lexdat index="5" token="belysas" id="8" db="general" length="1" mosy="VBPIM" >
                <lem>
                        <sym>belysa.vb</sym>
                        <lex><sym>belysa.vb.1</sym><val>va.plundra</val></lex>
                        <trglex><sym>enlighten.vb.1_id8</sym></trglex>
                </lem>
        </lexdat>
        <lexdat index="6" token="." id="9" db="general" length="1" mosy="SRA" >
                <lem>
                        <sym>stop.sr</sym>
                        <lex><sym>stop.sr.1</sym></lex>
                        <trglex><sym>stop.sr.1_id9</sym></trglex>
                </lem>
        </lexdat>
</s>
```

Figure 3.5: Output from the dictionary lookup

```
<s id="0" length="7"> <chart> <edge id="0" index="0" length="7">
  <phr.cat>sent</phr.cat>
  <first>
        <phr.cat>cl</phr.cat>
        <cl.type>main</cl.type>
        <mode>decl</mode>
          <verb>
            <head>
            <lex> <sym>komma.vb.1</sym> <val>va.komma</val> </lex>
            <trglex> <sym>come.vb.1_id5</sym> </trglex>
            <word.cat>VERB</word.cat>
            </head>
            <verb.type>main</verb.type> <tense>pres</tense>
            <inff>fin</inff> <diat>act</diat>
          </verb>
          <subj>
            <phr.cat>np</phr.cat>
            <df>
              <phr.cat>ng</phr.cat>
              <head>
                <lex> <sym>frågeställning.nn.1</sym> </lex>
                <trglex> <sym>issue.nn.1_id4</sym> </trglex>
                <word.cat>NOUN</word.cat>
              </head>
              <gender>utr</gender> <numb>plur</numb>
              <form>def</form> <case>basic</case>
              <sex>n</sex> <person>3</person>
              <attr>
                <phr.cat>adjp</phr.cat>
                <head>
                  <lex> <sym>etisk.av.1</sym> </lex>
                  <trglex> <sym>ethical.av.1_id2</sym> </trglex>
                  <word.cat>ADJ</word.cat>
                </head>
                <degree>pos</degree> <gender>nil</gender>
                <numb>plur</numb> <form>nil</form>
                <case>basic</case>
              </attr>
            </df>
            <gender>utr</gender> <numb>plur</numb>
            <def>def</def> <case>basic</case>
            <sex>n</sex> <person>3</person>
            <quant>
              <lex> <sym>den.al.1</sym> </lex>
              <trglex><sym>the.al.1_id0</sym></trglex>
              <word.cat>ART</word.cat> <gender>nil</gender>
              <numb>plur</numb> <form>def</form>
            </quant>
          </subj>
          <obj.inf>
            <phr.cat>cl</phr.cat>
            <cl.type>inf</cl.type>
            <inf.mark>
              <lex> <sym>att.ie.1</sym> </lex>
              <trglex> <sym>to.ie.1_id6</sym> </trglex>
              <word.cat>IE</word.cat>
            </inf.mark>
            <verb>
              <head>
                <lex> <sym>belysa.vb.1</sym> <val>va.plundra</val> </lex>
                <trglex> <sym>enlighten.vb.1_id8</sym> </trglex>
                <word.cat>VERB</word.cat>
              </head>
              <verb.type>main</verb.type> <inff>inf</inff>
              <diat>pass</diat>
            </verb>
          </obj.inf>
  </first>
  <sep>
          <lex><sym>stop.sr.1</sym></lex>
          <trglex><sym>stop.sr.1_id9</sym></trglex>
          <word.cat>SEP</word.cat> <type>major</type>
  </sep>
</edge> </chart> </s>
```

Figure 3.6: Output after transformation

3.5.1 Converting Tokens

In order to convert tokens, mapfiles that contain rules for transforming between dependency and feature structure representations were constructed. Three different mapfiles are used, one for converting part-of-speech tags and morphological features, one for converting dependency labels and one for inserting phrase nodes where they are needed.
The rules should be in the format

```
source|condition   ->    target
```

where *source* is the part-of-speech tag, morphological feature or dependency label given to a word by MaltParser and *target* is a corresponding set of features in MULTRA. The target feature set can contain multiple features, separated by underscores. This is useful since the categories used in MULTRA in many cases are more fine-grained than the ones assigned by the parser. The *condition* part of the rule is optional and is used for disambiguation in cases of one-to-many mappings.

The condition is an attribute value pair that is defined in the transformation component. A conversion from source to target tag can only take place if the condition holds for the token being processed or if the mapping has no condition. Take for example the rule for the present tense morphological feature. This feature is used for both verbs and participles in the MaltParser output, but in MULTRA should be transformed into two separate features depending on the word's part-of-speech category. Therefore the mapfile contains two rules for this tag, each with a condition specifying which type of word it applies to. The program will use the mapping for which the condition holds for the token in question.

```
PRS|pos:VB      ->      tense:pres_inff:fin
PRS|pos:PC      ->      part.type:pres
```

Conditions can be of several types, depending on the type of information they require to be checked. A *pos* condition requires a specific part-of-speech tag to be assigned to the word being processed in order for the rule to be active, as in the examples for the present tense feature above. A *feat* condition requires that the word in question has a feature with the value specified for it to hold, as in the rule for words tagged with *VG* for verb group. Only words with the *VG* tag and a feature tag with the value *SUP* for supine are affected by the rule.

```
VG|feat:SUP     ->      obj.sup_phr.cat:cl_cl.type:sup
```

The condition type *mosy* uses information extracted from the dictionary file for disambiguation. The *mosy* code is a code with morphological information used in Convertus' system. It contains information about word category and features in a compressed format and can be useful since it can contain information not present in the analysis given by MaltParser. For example it can be used for disambiguation in cases where one source part-of-speech category should be mapped into more than one target category. An example is the category *DT* for determiners, which in the annotation scheme used in Talbanken contains both articles and determinative pronouns. In MULTRA these words should be tagged as either pronouns or articles, which is done by checking their mosy codes and sorting out the words whose codes begin with *AL*, marking them as articles.

```
DT              ->      word.cat:PRON
DT|mosy:AL      ->      word.cat:ART
```

In some cases it is necessary to create very specific rules, concerning only a certain word or phrase. For those cases it is possible to use the word form itself as the condition, activating the rule only when this word is encountered. WH-pronouns, tagged with *HP*, should be treated as pronouns in MULTRA with the exception of the word *som*, which should be analysed as a subordinating conjunction. The rules below show how this exception is expressed in the mappings.

```
HP      ->      word.cat:PRON
HP|lex:som      ->      word.cat:CONJ_subju:+
```

Apart from using information about the token being processed, disambiguation can also be done by looking at the parent token, i.e the head of the dependency relation. The same condition types are used here, prefixed by the word parent. The rule below constructs a comparative unit for a word with the part-of-speech tag *AB* and a dependency parent that is the word *än*

```
UA|parentLex:än&&pos:AB         ->      comp.unit_phr.cat:advp
```

Rules are applied in an order going from more specific to more general, where rules without conditions are only applied if no more specific rules are found.

It is possible to construct a rule with an empty mapping, which simply removes the feature from the analysis. An example of this is the morphological feature *SMS*, that signals the presence of a compound. This information is not used by the translation system which treats compounds in the same way as other nouns, and therefore this feature can be safely removed from the analysis.

Converting Part-of-speech Tags and Morphological Features

The conversion of part-of-speech tags and morphological features is done in one of the transformation mapfiles. The mapfile contains rules that transform the part-of-speech tags in the input analysis into features with the attribute *word.cat* where the value is a part-of-speech tag in the MULTRA convention. For the finite verb *kommer* in the example sentence the following rule would apply:

```
VB      ->      word.cat:VERB
```

The file also contains rules for transforming morphological features into their MULTRA equivalents. The verb *kommer* has two features, as can be seen in the MaltParser output in figure 2.3. The features are *PRS* for present tense and *AKT* for active voice. The rules transforming them can be seen below.

```
PRS|pos:VB      ->      tense:pres_inff:fin
AKT     ->      diat:act
```

The rule for *PRS* contains two target features. The *inff* feature marks verbs as finite or infinite and is not a part of the tagset for the Talbanken data used to train MaltParser. In MULTRA however, it is mandatory for all verbs and so it is added to the analysis in this way.

Converting Dependency Labels

The second transformation mapfile contains rules for transforming dependency labels into MULTRA feature sets. The subject of the example sentence *De etiska frågeställningarna* consists of three words with the dependency labels *DT*, *AT* and *SS* (see figure 2.3). They are transformed using the following rules:

```
DT      ->      quant
AT      ->      attr_phr.cat:adjp
SS|pos:NN       ->      subj_phr.cat:np
```

The first rule transforms the determiner into the MULTRA syntactic category *quant*. The other rules transform the label *AT* into an attribute with the prase category *adjp* and the label *SS* into a subject with the phrase category *np*. We now have a noun phrase with an attribute and a quantifier, but it is not yet a complete transformation. Noun phrases in MULTRA need to have a description field *df*, that contains the head word and its attributes. Other modifiers such as determiners and quantifiers are not part of the description field.

The description field can not be a part of the target feature set for *SS* in the rule above, since it should be inserted at another level in the syntactic tree than the other features. Instead it is inserted by checking the third mapfile, that contains rules for inserting extra levels of syntactic analysis into the feature structure representation of a sentence. This mapfile contains the rule below, that tells us that if we find a nominal subject, we should insert a description field one level below it in the syntax tree. The description field gets the phrase category *ng*, noun group.

```
subj|pos:NN     ->      df_phr.cat:ng
```

The mapfile also contains rules for other cases where a subordinate level needs to be inserted into the syntactic tree in order to form a correct analysis in MULTRA.

3.5.2 Building Syntax

When all tokens have been transformed into MULTRA compatible structures the transformation component constructs the output XML document by concatenating tokens. While doing this, the component adjusts the syntactic analysis when needed, to make it compatible with MULTRA. The output is built top-down, starting with the root token of the sentence and then attaching all of its children and their children, until all tokens are concatenated. Depending on the part-of-speech category of the root token, a phrase category is given to the whole segment. For the example sentence in figure 3.6 the phrase category is *sent*, which means that we are dealing with a complete sentence, not a fragment. For this sentence, a subordinate level is inserted, consisting of a main clause.

The root token, the verb *kommer*, is attached to the main clause element. After that comes the noun phrase constituting the subject and finally the infinitive phrase *att belysas* which is analysed as an infinitive object.

Multi-word Expressions

For translation it is important to correctly recognise and process idiomatic expressions and phrase constituents containing more than one word. The Convertus Syllabus Translator dictionary contains entries for a lot of these expressions, and they are translated as one unit in Convertus' system. Since the input data is processed one token at a time by the transformation component, in order to use the lexical resources available for multi-word expressions these would have to be recognised and concatenated in the transformation process.

Recognition is done in pre-processing, when creating the dictionary file. A lookup of the input tokens is done in the Convertus Syllabus Translator dictionary, and if multi-word expressions are found the tokens are concatenated and presented as one unit in the lexicon file. The unit will be indexed by the first token in the sequence. When the transformation component appends lexical information to a token, it searches the lexicon file by the token index. When no token is found with the specified index, that is an indication that the word belongs to a multi-word expression. Since the first word in the expression already contains the lexical information for the whole expression, there is no need to attach this token to the syntactic analysis and so it will be skipped. All the children of this token will be attached to the first word in the expression instead.

The concatenated expression will get the syntactic label of the first word in the expression, which is not always a good analysis for the whole expression. One solution for correcting this is to look at the mosy code for the whole expression and try to set a new label with the help of the information found there. The program currently does this for prepositions and subordinating conjunctions, which get the labels *prep* and *introd* respectively. The assignment of labels for multi-word expressions is something that could be improved in future versions of the transformation component, enhancing it to handle other word categories and improving the label assignment to make it more accurate.

Reflexive Verbs and Verbs with Particles

Other types of multi-word expression are reflexive verbs and verbs with particles. However, they are handled by the grammar component in Convertus' translation system and so the same method applied to idiomatic expressions can not be used for them. The MaltParser trained on Talbanken analyses reflexive verbs as verbs followed by a direct object or, in some cases, an indirect object. The transformation program looks for occurences of verbs followed by an object with the word form *mig*, *dig*, *sig*, *oss* or *er* to find reflexive verbs. If such occurences are found they are concatenated into one token, analysed as a reflexive verb. Verb particles get the tag *PL* in the output from MaltParser and thus are easily identified and concatenated in the same way as reflexive verbs.

Identical Sister Nodes at Same Level

MaltParser does not place any restrictions on the occurence of words at the same level in the dependency graph with identical labels. A head word can have an unlimited number of dependents with the same label. This is a problem as in MULTRA all nodes need to be unique at a given level in the syntax tree. The occurence of identical sister nodes is mostly due to parsing errors, but sometimes they can be the result of a correct analysis, for example when concerning adverbs. I have chosen to solve this problem by simply enumerating all identical nodes when they occur, thus making them unique. This solution calls for some modifications to the transfer rules, where some default rules need to be inserted to handle these structures.

4 Results

In order to assess the quality of the translations produced by the MaltParser-based system, a comparison was made with the current Convertus system. Both systems were used to translate a test corpus, and the results were then compared to a gold standard translation of the corpus, performed by a human translator.

4.1 Test Setup

The data used for testing consists of 1971 text segments, taken from Convertus' translation memory. The segments have been extracted from course syllabi and thus consist of text from the domain being translated by the system in daily use. The use of this data will produce a favor for the Convertus system in testing, due to the fact that the text used to train MaltParser is from a different domain which might result in lower accuracy in the parsing step. The UCP parser, on the other hand, has been developed to work with this type of text and therefore has an advantage.

For technical reasons the test data was divided into eight evenly sized parts, that were run separately through the system. Test data part 1–7 thus consist of 250 segments each, while part 8 consists of 221 segments. Test results are reported for each data part separately as well as for the whole test corpus of 1971 segments.

The test corpus was translated by both systems and the results compared to the manual translation of the same data. For evaluation the BLEU (Papineni et al., 2002) metric was used.

4.2 Test Results

Before looking at translation quality, an examination is done of the results of running the test data segments through the MaltParser-based translation system developed in this thesis. There are a lot of obstacles along the way and a small mistake in the transformation process can result in no translation at all being produced. Thus, the question that needs to be answered first of all is if this system will manage to produce any translations for the transformed segments. The answer, after testing, is yes, for a small part of the test data. Of the 1971 segments in the test data, around 43 % were given a translation by the system. For the rest of the segments the system failed to produce a translation, due to errors during either the transfer or generation step of the translation process. A detailed result of all the tests is shown in table 4.1.

Test data	Segments	Translated	% translated
1	250	119	47,6%
2	250	108	43,2%
3	250	94	37,6%
4	250	90	36,0%
5	250	124	49,6%
6	250	105	42,0%
7	250	106	42,4%
8	221	106	48,0%
Total	**1971**	**852**	**43,2%**

Table 4.1: Number of successfully translated sentences

4.2.1 Translation Quality

Translation quality has been measured by calculating BLEU-scores for the eight test data files. Scores were calculated for the data translated by Convertus' UCP-based system as well as for the translations made by the MaltParser-based system. In this case the segments where translation failed were removed from the data. So the scores in table 4.2 are calculated for a subset of the testdata containing only segments successfully translated by the MaltParser-based system.

Due to errors in the transformation process, some of the segments that made it through transfer and generation in the MaltParser-based system have only recieved partial translations. This seems to mainly concern coordinations, where in some cases only the conjunction itself has been translated and the rest of the segment has simply been left out, leaving us with the translation *and* for a whole sentence. Naturally, this has a great effect on BLEU-scores, producing the significantly lower scores we see in table 4.2 for the MaltParser-based system compared with Convertus' UCP-based system.

To get an idea of how well the modified system could perform, given that the transformations from dependency to feature value structure are without errors, I have manually gone through the translated data, removing all partial translations and only keeping the ones where a segment has been fully processed. Both systems were tested again on this modified data, the resulting BLEU-score are given in table 4.3. As can be seen they are significantly improved and closer to the scores achieved by the UCP-based system.

4.2.2 Error Analysis

The errors made in the transformation from dependency to feature value structure result in a failure to find matching rules for the segment in question, in either the transfer or the generation stage of the translation process. The failed translations are due to errors in generation for the most part, with around one third of them being the result of transfer errors. An overview of all the

Test data	UCP-based system	MaltParser based system
1	0,3728	0,1982
2	0,3021	0,1484
3	0,5684	0,1009
4	0,9255	0,1741
5	0,4392	0,2877
6	0,3799	0,1255
7	0,3591	0,1380
8	0,4424	0,2370
Average	**0,4737**	**0,1762**

Table 4.2: Translation quality

Test data	UCP-based system	MaltParser-based system
1	0,4365	0,4171
2	0,5269	0,4532
3	0,6098	0,4060
4	0,9787	0,3734
5	0,5664	0,4809
6	0,5115	0,4228
7	0,4315	0,3850
8	0,5216	0,4088
Average	**0,5729**	**0,4184**

Table 4.3: Translation quality for fully translated sentences.

errors can be seen in table 4.4.

The most common transfer error in my data is segments that have got two end-of-sentence delimiters. This occurs for sentences where a minor delimiter has been made the root token of a sentence. The transformation program is implemented to discard all minor delimiters, while appending major delimiters to the top level of the segment being processed. If a minor delimiter is the root of a sentence the transformation program will treat it as major and produce a feature value structure that contains two delimiter nodes at the same level in the syntax tree, which is not allowed by MULTRA and leads to a transfer failure.

Generation errors have multiple reasons, the biggest problem being the handling of coordinations. The data contains a lot of enumerations, and these are sometimes difficult to handle for the transformation program. In some cases one of the conjuncts in a coordination have not received the dependency label *CJ* for conjunct by MaltParser, which leads to this word being left out when the coordinated phrase is built during transformation. There are also quite a lot of

Test data	Total errors	Generation errors	Transfer errors
1	131	93	38
2	142	97	45
3	156	99	57
4	160	78	82
5	126	94	32
6	145	105	40
7	144	108	36
8	115	83	32
Total	**1119**	**757**	**362**

Table 4.4: Error types

cases where a post attribute for one conjunct has been assigned the conjunction as head, which becomes a problem for the transformation program that fails to attach it to the correct phrase in the MULTRA structure.

5 Discussion

Testing shows that the system developed in this thesis can produce good quality translations, if the transformation between dependency and feature value structure is done correctly. In order to pass through the transfer and generation steps in the translation process, segments need to have the proper feature value structure, and even small deviations may lead to a failed translation. This is due to the fact that translation rules can be very specific, depending on certain features to be present in the input, and if they are absent no rule will be triggered.

Out of the 1971 segments used to test the system developed in this thesis, 43% got through both transfer and generation and received a translation. Out of these segments, another 25% were only partially translated, giving us a final of 32% fully translated sentences, which is a very modest amount. However, these 32% segments got good quality translations, resulting in a BLEU-score of 0.4184, which is close to the translation quality of the original Convertus system, with a BLEU-score of 0.4737.

There is still room for some improvement of the transformation module, dealing with errors like the ones described in section 4.2.2. Parsing and tagging accuracy could be improved by training on in-domain data, and parsing might also benefit from fine-tuning the features used in the parser module. However, it might also be necessary to make some adjustments to the MULTRA generation rules, in order to produce a system which can be fully functional.

One of the major issues in the transformation process concerns coordinations, which are very common within the domain of course syllabi. Typically the coordinations in course syllabi differ from those in ordinary text in that they are very long, often containing more than 10 conjuncts. These sentences have been problematic for MaltParser to handle, and the analysis of them is often inconsistent. It is difficult to handle these inconsistencies in the transformation program, but hopefully the problem could be solved by training MaltParser on in-domain data.

In the MaltParser-based translation system, no fallback strategies have been implemented for those cases where a segment fails to make it through the transfer and generation steps of the translation process. Since a 100% success rate for transformation between dependency and feature value structure seems highly unlikely, some sort of fallback strategy is much needed. Currently the system produces no output at all for segments that fail the transformation. A direct translation by simply looking the input words up in the dictionary would be one possible way of coping with these problematic segments.

The system tested in this thesis has been working with translations from Swedish to English, and one question is if it is possible to adapt it to handle multiple languages. This could be accomplished by replacing the mapfiles with ones following the annotation scheme in use for the language in question. The transformation program itself should not have to be altered.

6 Conclusion

The question this thesis has tried to answer is whether it is possible to integrate a dependency parser into a machine translation system operating on feature value structure by converting the parser output into the appropriate format. The test results show that this can indeed be an option, and that it is possible to achieve acceptable translation quality with this approach. However, the transformation program developed in this thesis is in its current state not good enough for use in a commercial system, since it only manages to produce translations for roughly a third of the input segments. With the proper improvements, in addition to some modifications of transfer and generation rules, I foresee that the system can become functional. Further improvement could be achieved by training the tagger and parser on in-domain data.

Bibliography

Björn Beskow. Unification-based transfer in machine translation. Ruul 24. Uppsala University., 1993.

Jan Einarsson. Talbankens skriftspråkskonkordans, 1976a. Lund University, Department of Scandinavian Languages.

Jan Einarsson. Talbankens talspråkskonkordans, 1976b. Lund University, Department of Scandinavian Languages.

Péter Halácsy, András Kornai, and Csaba Oravecz. Hunpos - an open source trigram tagger. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, 2007.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*, chapter 15. Pearson Education, Upper Saddle River, New Jersey, 2 edition, 2009.

Joakim Nivre. *Inductive Dependency Parsing*, volume 34 of *Text, speech and language technology*. Dordrecht:Springer, 2006.

Joakim Nivre, Jens Nilsson, and Johan Hall. Talbanken05: A swedish treebank with phrase structure and dependency annotation. In *Proceedings of LREC*, pages 1392–1395, 2006.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 2007.

Kishore Papineni, Salim Roukos, and Todd Ward. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*. Microtome Publishing, Brookline, Massachusetts, 3 edition, 2003.

SUC. Department of linguistics, umeå university and department of linguistics, stockholm university, 1997. SUC 2.0 Stockholm Umeå Corpus, Version 2.0.

Anna Sågvall Hein. Language control and machine translation. In *Proceedings of the 7th International Conference on Theoretical and Methodological Issues in Machine Translation*, St. John's College, Santa Fe, New Mexico, July 1997.

Anna Sågvall Hein. An experimental parser. In *Proceedings of the 9th International Conference on Computational Linguistics (Coling 82)*, Prague, 1982.

Anna Sågvall Hein, Eva Forsbom, Per Weijnitz, Ebba Gustavii, and Jörg Tiedemann. MATS - a glass box machine translation system. In *Proceedings of the Ninth Machine Translation Summit*, pages 491–493, New Orleans, USA, September 2003.

Per Weijnitz. Uppsala chart parser light. improving efficiencey in a chart parser. Computational linguistics. language engineering programme, Uppsala University, 2002.

Per Weijnitz, Anna Sågvall Hein, Eva Forsborn, Ebba Gustavii, Eva Pettersson, and Jörg Tiedemann. The machine translation system MATS - past, present & future. In *Proceedings of RASMAT'04*, Uppsala, Sweden, April 2004.