



UPPSALA  
UNIVERSITET

# **Intra-sentence segmentation as a pre-processing step in machine translation**

Anton Eriksson, Tora Myhrman

Uppsala University  
Department of Linguistics and Philology  
Språkteknologiprogrammet  
(Language Technology Programme)  
Master's thesis in Computational Linguistics  
June 7, 2010

Supervisors:  
Jörg Tiedemann, Uppsala University  
Eva Pettersson, Convertus AB

## Abstract

This thesis presents a method for segmenting Swedish sentences as a pre-processing step for rule-based or hybrid machine translation. The purpose is to reduce translation time.

Sentences are split based on rules for contexts that include conjunctions, subjunctives and delimiters. Rules have been developed for three different types of input. In the rules for chunked data, string patterns and tag patterns as well as chunk information are used to match sentences. In the rule set for tagged sentences, tag matching and string matching are used. In rules for non-tagged sentences, string-matching is used.

We have evaluated the segmentation with data from three different domains. Results show consistently reduced processing times, at best close to 30 percent. The best results were achieved using tagged data. Initially we had hoped that the translation quality would also improve, because of the simplified structure of the segments compared to the original sentences. This was not the case. BLEU scores for the segmented data and the non-segmented data are similar.

## Sammandrag

I den här uppsatsen presenteras en metod för att segmentera svenska meningar som ett förprocessningssteg för regelbaserad eller hybrid maskinöversättning. Syftet är att minska översättningstiderna.

Meningar delas baserat på mönster för kontexter som innehåller konjunktioner, subjunktioner och skiljetecken. Reglerna har utvecklats för tre olika typer av indata. I reglerna för chunkad data används strängmönster, taggmönster och chunkinformation för att matcha meningar. I reglerna för taggad data används strängmönster och taggmönster. I reglerna för data som inte taggats eller chunkats används strängmatchning.

Vi har utvärderat segmenteringen med data från tre olika domäner. Resultaten visar på en konsekvent minskning av översättningstider, som bäst nära 30 procent. Bäst resultat uppnåddes för taggad data. Till en början hade vi även hoppats på en ökning av översättningskvaliteten, tack vare segmentens förenklade struktur jämfört med de osegmenterade meningarna. Så var inte fallet. BLEU-värdena för den segmenterade datan och den osegmenterade datan är snarlika.

# Contents

<b>Abstract</b>	<b>2</b>
<b>Sammandrag</b>	<b>2</b>
<b>Acknowledgements and division of work</b>	<b>7</b>
<b>1 Introduction</b>	<b>8</b>
1.1 Purpose . . . . .	8
1.2 Convertus . . . . .	9
1.3 Motivations for this project . . . . .	9
1.4 Outline of this thesis . . . . .	10
<b>2 Background</b>	<b>11</b>
2.1 Machine translation . . . . .	11
2.2 Convertus' translation system . . . . .	12
2.2.1 Multra . . . . .	13
2.2.2 UCP . . . . .	13
2.3 Tagging and HunPos . . . . .	14
2.4 Chunking and Swe-SPARK . . . . .	15
2.5 Evaluation methods and metrics . . . . .	15
2.5.1 Human evaluation . . . . .	16
2.5.2 Automatic evaluation . . . . .	16
2.5.3 BLEU . . . . .	16
2.5.4 METEOR . . . . .	18
2.5.5 TER . . . . .	18
2.6 Previous work . . . . .	19
2.6.1 Sentence segmentation for machine translation . . . . .	19
2.6.2 Other pre-processing methods for machine translation . . . . .	20
2.6.3 Sentence segmentation and text simplification . . . . .	21
<b>3 Implementation</b>	<b>24</b>
3.1 Segmentation principles . . . . .	24
3.2 The program . . . . .	26
3.2.1 Rule file syntax . . . . .	27
<b>4 Segmentation rules</b>	<b>28</b>
4.1 Rejected rules . . . . .	28
4.2 Rules for tagged and chunked sentences . . . . .	30
4.3 Rules for tagged sentences . . . . .	31

4.4	Rules for non-tagged sentences . . . . .	31
<b>5</b>	<b>Evaluation</b>	<b>33</b>
5.1	Method . . . . .	33
5.2	Data . . . . .	33
5.3	Minimum sentence lengths . . . . .	35
5.4	Results . . . . .	36
5.4.1	Analysis of example sentences . . . . .	39
<b>6</b>	<b>Discussion</b>	<b>43</b>
<b>7</b>	<b>Conclusion</b>	<b>46</b>
	<b>Bibliography</b>	<b>47</b>

## List of Figures

2.1	Flowchart of Convertus' translation process . . . . .	12
2.2	Illustration of the lexicon hierarchy used by the syllabi translator .	13
3.1	Algorithm . . . . .	26
5.1	Results for different minimum sentence lengths . . . . .	36

# List of Tables

1.1	Average sentence lengths and translation times for non-segmented data from different domains . . . . .	10
5.1	Data properties . . . . .	35
5.2	Translation quality for the different data sets and rule sets . . . . .	36
5.3	Number of sentences segmented for the different domains . . . . .	37
5.4	Syllabi data results . . . . .	37
5.5	Patent application data results . . . . .	38
5.6	Technical manuals data results . . . . .	38

## Acknowledgements and division of work

We have worked closely together on all parts of this thesis. However, we have divided some parts of the work between us. In the Background chapter, Anton has made the research on, and written the parts about, Tagging and HunPos 2.3, Chunking and Swe-SPARK 2.4, UCP 2.2.2, Evaluation methods and metrics 2.5, Human evaluation 2.5.1, Automatic evaluation 2.5.2 and BLEU 2.5.3. Tora has researched and written about Machine translation 2.1, Convertus' system 2.2, Multra 2.2.1, METEOR 2.5.4, TER 2.5.5 and Previous work 2.6. During implementation Tora wrote the Java program. Anton wrote the shell scripts and Perl programs used to prepare the data, re-attach the segments and evaluate and give metrics for comparing data. Anton also formulated the rules into regular expressions.

We would like to thank our supervisors Jörg Tiedemann and Eva Pettersson for great ideas, advice and support. We would also like to thank Per Weijnitz at Convertus AB for technical support, and Anna Sågwall Hein for tips and ideas. Also, many thanks to Sebastian Wiberg for advice on how to optimise our code and to Ida Köhler for proof-reading this thesis.

# 1 Introduction

“Part of the reason why translation is difficult for computers is that translation is just difficult: difficult even for humans.” (Arnold, 2003, p. 119)

Of course, machine translation has other challenges than a human translator as well. A computer can follow rules or learn context patterns from training data but it cannot improvise or make sense of anything unexpected. Computers can store sentences and their translations in translation memories (TMs), but the productivity of language creates a need for analysis and translation of sentences never encountered before. The complexity of language, and the fact that it is possible to create grammatical sentences of arbitrary length, can make this analysis strenuous on time and memory resources.

Li et al. (1990, p. 410) define a long English sentence, from the parsing point of view, as “a sentence which has complicated syntactic structure or has too many words in it”. They segmented English sentences as a preparatory step for machine translation because they noticed that translation times grew substantially for long sentences. Parsing time increased exponentially with increased branching, reflecting complicated syntactic structure, and search depth, reflecting sentence length. (Li et al., 1990, p. 410).

Besides potentially complicating parsing, long sentences are less likely to trigger matches in translation memories. Craniias et al. (1994) examined the possibilities of matching sub-sentence text units using similarity clustering. They concluded that the system performed better with smaller text units and larger number of clusters. But, “on the other hand if the text unit is the sub-sentence we face one major problem, that is the possibility that the resulting translation of the whole sentence will be of low quality, due to boundary friction and incorrect chunking” (Craniias et al., 1994, p. 100).

## 1.1 Purpose

Our purpose is to examine what effects initial segmentation of long sentences may have on the efficiency and quality of rule-based or hybrid machine translation. We will evaluate different combinations of PoS-tagging, chunking and string splitting on key words to identify clauses in sentences. Our aim is to find the method most suitable for sentence segmentation when used as a pre-processing step in machine translation. The translation system we will use is the Convertus translator, which is a hybrid system using translation rules as well as translation memory and statistical fallback methods.

We will experiment with different minimum sentence lengths to find out above which length sentences benefit from segmentation. We will translate

both segmented and non-segmented versions of the source text using Convertus' system and compare the translations to evaluate the effect on translation quality and processing time.

Our hypothesis is that the segmentation will make the translations go faster. Translation times will be reduced by simplifying the input because there will be less need for the time-consuming statistical fallback methods. Also, translation quality may improve, because the transfer, parsing and generation will be working with less complex sentences. The shorter segments may also be more likely to trigger translation units in a translation memory.

There is a risk that the segmentation ruins the syntactic structure, making it impossible to correctly parse the resulting fragments. We need to find safe places to segment the sentences, so that translation quality is not negatively affected by the fact that the sentences have not been analysed and translated as a whole.

The source language is Swedish and the target language is English. The texts used are from three different domains: syllabi, patent applications and technical manuals.

## 1.2 Convertus

Convertus is a machine translation company that started out as a spin-off from Uppsala University. They offer both rule-based and statistical methods of machine translation, and adapt the system to fit the clients' domains.

Convertus also supplies a spell checker, based on the same lexicons as the translator. The fact that the same lexicons are used creates consistent spelling, making sure that words in the translation lexicon are actually recognised. However, users do not always utilise the spell checker which means that spelling errors might remain.

## 1.3 Motivations for this project

For Convertus, problems with long and complicated sentences became an issue when handling translations of patent applications. These texts have many long sentences and were in some cases not even possible to translate because processing times became too extensive.

Our data comes from three different domains, with varying average sentence length and complexity. Table 1.1 shows the average sentence lengths in number of words, and the average translation times per sentence, for data that has not been segmented. There is a substantial increase in average translation time for the sentences in the patent applications, which is the domain with the longest and most syntactically complex sentences (see section 5.2 for examples and more information on the data).

The differences in sentence length and complexity is of course not the only explanation for the increased translation times. There are other important aspects to bear in mind, the most crucial being that the patent data is translated without a domain-specific lexicon (see chapter 5 for more information).

**Table 1.1:** Average sentence lengths and translation times for non-segmented data from different domains

<b>Data</b>	<b>Average sentence length (No. of words)</b>	<b>Average sentence translation time (seconds)</b>
Syllabi (727 sentences)	12.9	3.54
Patent applications (312 sentences)	23.6	27.23
Technical manuals (511 sentences)	13.6	3.45

## 1.4 Outline of this thesis

In chapter 2 we describe the methods and tools used during our work. We also present previous work with relation to sentence segmentation and machine translation pre-processing.

The next chapter, 3, describes our approach to implementing a sentence segmenter, including the principles for segmentation and the program. Chapter 4 lists and explains the rules used. Chapter 5 presents the methods for evaluation and the data, as well as the results of the evaluation in section 5.4.

Finally, we discuss the results in chapter 6 and give some concluding remarks and thoughts for the future in chapter 7.

## 2 Background

This chapter presents the resources used during the development of our sentence segmentation tool and gives an academic background by presenting a selection of related research. First, machine translation in general and the machine translator used for this work are presented. Then the tagging and chunking methods utilised are described. The methods chosen are suitable for Swedish, since this is the source language used. The section thereafter deals with methods of evaluation. Finally, related work is presented.

### 2.1 Machine translation

Machine translation (MT) is “the use of computers to automate translation from one language to another” (Jurafsky and Martin, 2009, p. 895). It is not one pure research field but “applied” research of different areas, like linguistics, computer science, artificial intelligence and translation theory (Hutchins and Somers, 1992, p. 3).

Rule-based machine translation (RBMT) works by applying rules to transform the source text to create a translation. Statistical machine translation (SMT) is usually phrase-oriented (Jurafsky and Martin, 2009, p. 913) and uses a language model to compute the probability of a phrase appearing in the target language, and a translation model to compute the probability of the target phrase being generated by the source phrase. According to Jurafsky and Martin (2000, p. 820) RBMT tries in every step to transform the input sentence into a fluent sentence in the target language, without altering the original meaning too much. This is different from SMT, where the task is to find the translation that maximises the product of fluency and faithfulness.

Hutchins and Somers (1992, p. 4) describe three different basic types of RBMT. The simplest and oldest method is direct translation, which only works with specific language pairs and uses a minimal amount of analysis. A more complex type is interlingua translation, which handles translation from source language to target language in two steps. An intermediate representation of the meaning of the source text is created, and the translation is generated from this representation. The third basic type is transfer. Transfer methods usually work in three steps. The first step is to syntactically and/or semantically analyse the source text, the second step is the transfer, and in the third step the text is generated using syntactic rules for the target language. Ambiguities in the source text are solved in the first step, and lexical and structural differences between languages are handled in the transfer step.

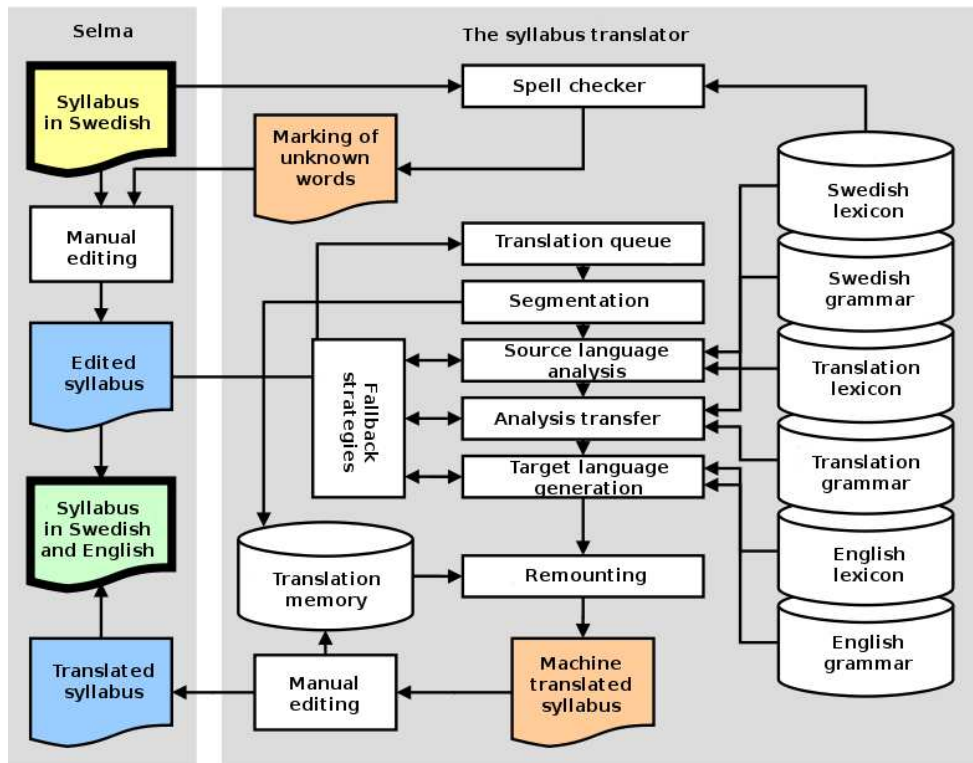


Figure 2.1: Flowchart of Convertus' translation process (Convertus)

## 2.2 Convertus' translation system

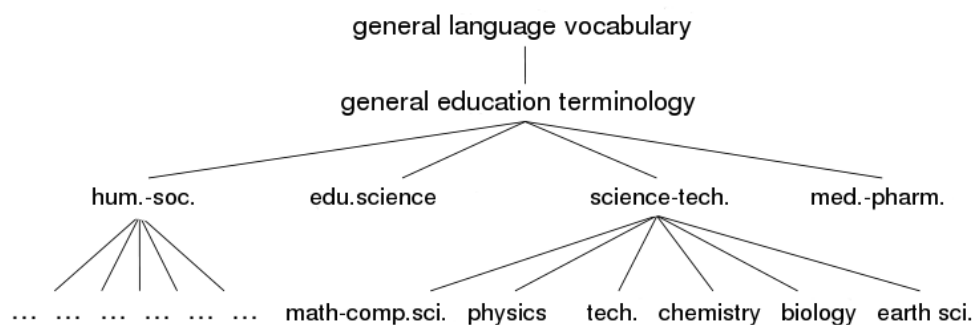
Convertus' translator is a Swedish-English machine translation system, originally based on the rule-based MATS system, developed at the Department of linguistics at Uppsala University (Weijnitz et al., 2004). The system consists of several modules where each of them handles different steps in the translation process. Figure 2.1 shows the different steps of the syllabi translator when integrated in the syllabi database system Selma. The core of the system is the translation engine Multra (see section 2.2.1) which is used for transfer and generation.

The input is handled sentence by sentence. The system uses three different types of lexicons. The source language and target language lexicons include lemmas and morpho-syntactic information, and the translation lexicon includes lexeme-to-lexeme translations. The system uses a core of general grammar and lexicons, and client-specific lexicons are added to fit different domains, as illustrated in figure 2.2.

The processing steps are<sup>1</sup>:

1. Extraction of sentences
2. Tokenisation
3. Tagging using HunPos (see section 2.3)

<sup>1</sup>Eva Petterson CEO Convertus AB, conversation February 10th 2010.



**Figure 2.2:** Illustration of the lexicon hierarchy used by the syllabi translator (Convertus)

4. Lexicon matching including compound analysis
5. Parsing using UCP (see section 2.2.2)
6. Transfer using Multra
7. Generation using Multra. If this generation fails, statistical methods (in form of a language model extracted from a corpus) are used.
8. Post-processing. Two methods are used; statistical edits based on earlier changes made manually by the users, and rule-based edits based on string-substitutions.

## 2.2.1 Multra

The transfer in Multra works with feature structures and is unification-based. Unification is “an operation that makes two feature structures identical” (Beskow, 1993, p. 11). If two feature structures contain any incompatible information, they cannot be unified.

The transfer-rules may be used to handle lexemes as well as structural units or idioms (Beskow, 1993, p. 12). An arbitrarily large context may be covered by a rule, since the rules relate to feature structures and not to, for example, parse trees (Beskow, 1993, p. 18). When several rules are applicable a more specific rule is preferred over a less specific.

The generation in Multra is unification-based as well, combining context-free grammar and feature structures (Beskow, 1997, p. 14). The generation is in fact a transformation from a complex functional representation of a sentence to a linear realisation of that sentence in the target language. Multra performs this generation in three separate steps: syntactic generation, morphological generation and phonological generation (Beskow, 1997, p. 5).

## 2.2.2 UCP

The Uppsala Chart Processor is a procedural, non-deterministic chart parser (Weijnitz, 2002). The parsing strategy of the processor is defined by the grammar (in contrast with declarative grammars) (Weijnitz, 2002, p. 6). Given an

input string to parse, the processor initialises a chart that is to be built from the content of a grammar and a dictionary. The chart structure is a graph built by vertices and edges. The edges are either passive or active. Passive edges span vertices that have been successfully processed, while active edges are still in process. The vertices represent the units of a given input string.

Parsing is carried out through a series of tasks that need to be completed. A task may arise when an active edge reaches a node from where a passive edge starts. When a task arises it is sent to a schedule that controls the order in which to carry out the tasks. Processing of a task may lead to new edges, activating new tasks. The parsing is completed when the schedule is empty, which happens when all tasks have been processed. Because all that is needed for the parsing to be completed is that all tasks have been processed, all parsing results may not span the entire input.

After producing all possible parsings results, the most suitable is chosen based on preference rules. If no result has successfully parsed the entire input, the longest spanning partial parsing result is extracted (Weijnitz, 2002).

## 2.3 Tagging and HunPos

Part-of-speech (PoS) tagging is the process of connecting words to appropriate PoS information. PoS-tagging is normally used for pre-processing purposes. The work of a tagger usually consists of assigning an appropriate part-of-speech tag to every token in a given input before sending the output (the tagged tokens) to, for instance, a parser. It is usually preceded by a tokenisation process, although the tokenisation may also be woven into the tagging process.

The different tags are defined by a tagset. Tagsets commonly used for Swedish are the SUC tagset (Ejerhed and Källgren, 1997) and the PAROLE tagset (Ejerhed and Ridings, 1997). For English, the Penn-Treebank tagset (Bies et al., 1995) is widely used. Apart from differing in what language they are suited for, the different tagsets are of varying complexity. They can range from only containing the most important word classes (verbs, nouns, etc.) to separating most characteristics possible to identify. The PAROLE tagset, consisting of 156 tags, is the tagset used by Convertus and therefore the tagset used for our work.

There are several different approaches to part-of-speech tagging. Almost all of them use some kind of machine learning algorithm and can be used for any language, as long as there is access to training data. One of the most well-used implementations is Trigrams'n'Tags (TnT), first presented by Brants (2000). TnT is a statistical part-of-speech tagger based on Hidden Markov models (HMMs). Other approaches include Toutanova et al.'s (2003) dependency network-based tagger, Ratnaparkhi's (1996) MaxEnt-based tagger, Brill's (1995) Transformation-Based Learning tagger and the SVM-based tagger created by Giménez and Márquez (2003).

HunPos (Halácsy et al., 2007) is essentially an open source reimplementa-tion of TnT. It is HMM-based and uses trigram language models. Other than being open source, it differs from TnT by using a one word context window to estimate lexical/emission probabilities (Megyesi, 2009). Unknown words are handled by a guessing algorithm, where tag probabilities are set by performing

a suffix analysis and looking at the words in the training data that share the same suffix (Halácsy et al., 2007, p. 210). To speed up processing, the Viterbi algorithm with beam search is used (Brants, 2000, p. 227).

## 2.4 Chunking and Swe-SPARK

Chunking is a type of shallow parsing. The purpose of chunking is to classify a given sentence's non-overlapping segments in a linguistically motivated way, typically using the most basic parts of speech to classify the segments. These classes are commonly the ones used to annotate content words and their phrases, but often a chunking task only requires identifying and classifying noun phrases. This means that not all words of a given sentence need to be classified by the chunking process (Jurafsky and Martin, 2009, p. 485). The two main approaches to chunking are finite-state rule-based chunking and machine learning-based chunking (Jurafsky and Martin, 2009, p. 486-489). These approaches are comparable to their corresponding full parsing approaches.

Swe-SPARK is a chunking application for Swedish (Megyesi, 2002). It uses a context-free grammar for Swedish and an implementation of Earley's algorithm programmed by John Aycock (1998). To be able to classify segments it requires the input text to be tokenised and tagged with PAROLE tags (Ejerhed and Ridings, 1997).

## 2.5 Evaluation methods and metrics

All scientific experiments need to be evaluated. This, of course, also applies to machine translation. There are several different methods of evaluation for machine translation output, but they all fall under two categories: human evaluation and automatic evaluation.

Automatic evaluation methods are meant to emulate human evaluation, but are less time-consuming. They are also more consistent, which can be both positive and negative, since acceptable alternative translations always exist.

It is difficult to determine what the perfect translation is, as it may be subjective. A translation can be exact and correct, even if a human translator might phrase it differently. It is vital to make explicit the criteria for manual evaluation of machine translated output, to give as little room as possible for subjectivity. This is even more important when applying these criteria to an automatic system, because a computer can not address any subjective judgement issues that may arise from unclear criteria.

Quality is not the only important evaluation factor for translation. Efficiency, regarding speed and cost, is as crucial (ALPAC, 1966) and is part of the motivation for machine translation in the first place. Quality and efficiency are of course related. Higher quality means less need for costly manual post-editing.

### 2.5.1 Human evaluation

Human evaluation of MT systems is the starting point of determining translation quality. Translations of this kind are meant for humans to read, which makes human evaluators the natural choice. At its core the evaluation process is very simple.

The first step is to decide on some criteria for determining quality. These may differ somewhat depending on what the translations are to be used for, but they should always be exact to minimise the subjectiveness of the evaluations. Different terms may be used for the criteria, but mainly translations are being judged for their fidelity and their fluency (Jurafsky and Martin, 2009, p. 930).

To have such broad criteria be exact, subdivisions are needed. For fidelity, these can pertain to how much of the information in the source is still available in the target and how accurate the terms in the target are for the domain of the source (Joscelyne, 2008). Fluency can be interpreted in various ways, but is usually defined as something along the lines of how clear and natural the translation seems and the appropriateness of the style (Jurafsky and Martin, 2009, p. 930). More generally, it can be said that translations are judged by how close they are to the original text.

### 2.5.2 Automatic evaluation

Most, if not all, automatic evaluation algorithms have been developed on the premise that the algorithm should, as closely as possible, reflect human evaluation. The reason for creating an automated system for evaluation is that a computer is almost certainly more efficient than a human in terms of time and, because of that, cost.

Even if the starting point is to emulate a human expert as closely as possible there may be another upside to automatising evaluations, besides saving time: an automated system is less capable of making nuanced judgements, which might lead to less accurate results but also to a more consistent evaluation. A less correct scoring is of course not desired, but if the errors are consistent there is usually more hope for amending them.

For the system to understand what is considered a correct translation of a text, it needs a gold standard. The gold standard consists of a number of manually translated sentences. As there can be more than one correct translation of a sentence, most MT metrics need multiple human translations for every sentence used in a test set (Jurafsky and Martin, 2009, p. 931). Building a test set demands a very large amount of manually translated data, but the sets are of course reusable.

### 2.5.3 BLEU

Possibly the most prominent automatic evaluation algorithm is BLEU (Bilingual Evaluation Understudy). The BLEU evaluation system uses a metric for determining how close a translation generated by an MT system is to a translation made by a human expert, and thus requires a corpus of reference translations produced by human experts (Papineni et al., 2002). The metric used for determining translation closeness is a modified n-gram measure for calculat-

ing precision. The modification in this case is that it penalises translations that overgenerates some of the words, even if the words themselves are correct.

First the system counts how many times a word is allowed to exist in the translation, by adopting the highest number of occurrences for that word in any of the relevant manual translations. That number is then divided by how many times the word occurs in the translation. Consider a machine translation that only contains words found in any of the relevant manual translations, but also only contains one single word over and over again. Its modified unigram precision score will be 1 divided by the highest amount of times that one word occurred in the manual translation which had the most occurrences of that word. The system will perform this calculation for every word in the MT output. Then, an average for every sentence in the output, given the modified precision for every 1-gram, is calculated. Finally, the system calculates an average for the entire output, given the average modified precision for every sentence. To take word order into account, longer n-grams than 1-grams will need to be taken into consideration.

The modified n-gram precision for an entire test corpus  $p_n$  is calculated according to the following formula (Papineni et al., 2002, p. 313)<sup>2</sup>:

$$p_n = \frac{\sum_{c \in \{Candidates\}} \sum_{n\text{-gram} \in c} Count_{clip}(n\text{-gram})}{\sum_{c' \in \{Candidates\}} \sum_{n\text{-gram}' \in c'} Count(n\text{-gram}')}. \quad (1)$$

With modified n-gram precision alone it is still possible for poor translations to receive a high score. For instance, a ten word long sentence that is translated into only two words will receive a perfect score if those two words are correct. To deal with this, BLEU incorporates a sentence brevity penalty. This is an evaluation measure used for making sure that the length of the sentences in the MT output matches the length of the manually translated sentences in the test set. The ideal score for the sentence brevity is 1.0. The output sentence will receive that score if it is as long or longer than the shortest corresponding sentence in the test set, but not longer than the longest one (Papineni et al., 2002, p. 315).

Let  $c$  be the length of the candidate translation and  $r$  the length of the effective reference corpus, and the brevity penalty  $BP$  is computed according to (Papineni et al., 2002, p. 315):

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}. \quad (2)$$

The final BLEU score for a given MT output can be between 0 and 1, 1 being a perfect score and 0 being an entirely incorrect translation. The score is calculated by multiplying the modified n-gram precision for the entire corpus with the brevity penalty (Papineni et al., 2002, p. 315):

$$BLEU = BP \times \exp\left(\sum_{n=1}^N w_n \log p_n\right). \quad (3)$$

---

<sup>2</sup> $Count_{clip}$  is the count of each n-gram, if needed truncated to not exceed the largest count of that n-gram in any of the reference translations.

Criticism against relying too much on BLEU scores has been based on the fact that BLEU scores do not always correlate with human judgements of quality. Statistical machine translation systems are often tuned to improve BLEU-scores. This makes the bias towards n-gram matches too large which may diminish the correlation with human judgement. Callison-Burch et al. (2006) found concrete examples of sentences that gained similar BLEU scores but were judged quite differently by human evaluators, both with respect to fluency and adequacy. They blame this partly on BLEU's lack of identifying synonyms and also on the fact that all words, content-bearing or not, are judged to be equally important. They also claim that BLEU allows too much variation in the order of which the n-grams occur (Callison-Burch et al., 2006, p. 251). Also, the sentence brevity can not fully compensate for the fact that recall is not a part of the calculation (Callison-Burch et al., 2006, p. 252). Other automatic measures have been suggested to make up for these problems. We will, in addition to BLEU, use two of them: METEOR (see section 2.5.4) and TER (see section 2.5.5).

## 2.5.4 METEOR

METEOR is an evaluation metric which aims to correlate with human judgement on a sentence level (Banerjee and Lavie, 2005). Words in the translation are mapped to the reference translation in three different steps. The algorithm begins by trying to find exact word matches. Then a stemmer is used to find the word stems and map them. Finally, words are mapped using information on synonymity found in WordNet (Miller and Fellbaum, 2007). When all possible word matches have been found, the largest word alignment subset is chosen. A parameterised f-score is computed from the unigram precision (P) and recall (R) values (Lavie and Agarwal, 2007, p. 229):

$$F_{mean} = \frac{P \times R}{\alpha \times P + (1 - \alpha) \times R} \quad (4)$$

A penalty for word order is added and the METEOR score for the two strings is calculated according to (Lavie and Agarwal, 2007, p. 229):

$$\text{score} = (1 - \text{penalty}) \times F_{mean} \quad (5)$$

The parameter used to compute the f-score and the values used to compute the penalty may be tuned to optimise the system, for example to better suit different languages (Lavie and Agarwal, 2007, p. 229). The optimisation is carried out to correlate as much as possible with human judgements.

Banerjee and Lavie (2005) showed that METEOR correlated better with human judgements than BLEU and ascribed this strongly to the fact that recall is a part of the equation.

## 2.5.5 TER

Snover et al. (2006) propose Translation Edit Rate (TER) as a more intuitive way of calculating the quality of translation. TER measures the numbers of editing operations needed to make the output sentence identical to a reference

sentence. The possible edits are shifts (of single words or word sequences), insertions, deletions and substitutions. A correct translation gets a TER score of 0 and a bad translation gets a score closer to 1. The calculation is performed according to (Snover et al., 2006, p. 3):

$$TER = \frac{\# \text{ of edits}}{\text{average \# of reference words}} \quad (6)$$

The algorithm begins by trying to find the optimal number of shift operations, that is: the number of shifts that gives the lowest number of total operations. Then the minimum number of deletions, insertions and substitutions is calculated. There is no weighting; all operations adds 1 to the number of edits. TER correlates reasonably well with human judgements but tends to overestimate the actual error rate. TER also correlates well with BLEU (Snover et al., 2006, p. 8).

## 2.6 Previous work

Segmentation, or simplification, of sentences may be a way of enhancing the performance of NLP applications since it allows some control of the input. Methods include, for example, sentence segmentation, clause segmentation and text simplification.

Some research has been made to find methods of pre- and post-processing specifically suitable for machine translation. Many translation systems work as black box systems, which means that the actual translation process is hidden from, and not affectable by, the user. This fact makes it suitable to try to improve the translation quality by applying changes to the texts prior to and after the translation itself (Aranberri Monasterio, 2010, p. 28).

“Pre-processing encompasses any method used to manipulate the source text to suit the MT system better” (Aranberri Monasterio, 2010, p. 28) and these methods include limiting the language variety of the source text, for example by using only domain specific texts or controlled language. Controlled languages (CLs) limit the lexicon and syntax allowed. The main reason to use a CL is to reduce the language ambiguities. CLs can improve the quality of translation because the source text becomes less complex and the lexicon more covering. Also, the higher level of consistency between texts makes it more likely to find matching units in a translation memory (Aranberri Monasterio, 2010, p. 30). Another common method used for pre-processing is compound splitting.

In post-processing, the MT output is manipulated to obtain a certain quality standard (Aranberri Monasterio, 2010, p. 43). This is done automatically as well as manually. Since RBMT uses a limited set of rules, the errors after translation are usually regular which makes it possible to correct many of them automatically, using a set of post-processing rules.

### 2.6.1 Sentence segmentation for machine translation

Li et al. (1990) tried to match long sentences with patterns to speed up parsing and hence machine translation. If a sentence matched one of the patterns it

was recursively partitioned according to pattern rules, and then segments were parsed both separately and combined. They managed to match about one third of the long sentences and concluded that this usually made the failed parses affect fewer words, but that time was wasted by the fact that parses were produced that, in the end, did not fit the input sentence (Li et al., 1990, p. 3).

Gerber and Hovy (1998) had the hypothesis that machine translation quality would be improved by segmenting sentences. They claimed that shorter sentences should be easier to process because there would be fewer attachment points for the constituents and therefore fewer ambiguities and fewer possible analyses to choose from (Gerber and Hovy, 1998, p. 451). They tested their hypothesis by segmenting Japanese sentences before translating them to English. The segmentation was rule-based and the criteria for sentences to be split were the following (Gerber and Hovy, 1998, p. 454):

- The original sentence is a minimum of 20 words long.
- Either: a continuative or infinitive form verb phrase is found followed by a comma, or: a clause conjunction is found.
- The verb phrase is not functioning as an adverbial/extended particle (e.g. *ni tsuite* ('concerning')).
- The resulting new sentence will be at least 7 words long.

Where necessary, the split sentences were supplied with a replacement subject and/or the verb tense was corrected. They used human evaluation to judge readability. Results were unfortunately not convincing.

Kim et al. (2000) used a statistical method for assigning probabilities to each potential segmentation of a sentence and choosing the most likely one. Because of sparse data, the machine learning was combined with manually defined word sets including, for example, conjunctions. This way they were able to segment most sentences and substantially improve parsing efficiency. Their method was used for idiom-based translation, since this is the most common method when translating from English to Korean because of the languages' large structural differences (Kim et al., 2000, p. 164). The target sentences' quality was not evaluated.

## 2.6.2 Other pre-processing methods for machine translation

Other ways of manipulating the source text to improve the translation have been tried. To change the word order is a common pre-processing method for SMT. The goal is to make the word order of the source sentence more similar to the word order of the target language. Xia and McCord (2004) used a hybrid system with automatically learned patterns and rule-based parsing to rewrite French sentences before translating them to English and managed to improve BLEU scores by ten percent. Collins et al. (2005) applied rules to parsed German sentences to re-order the phrases before translating them to English and improved the translation quality, according to both BLEU scores and judgements made by human evaluators.

Babych et al. (2009) used re-writing rules to change often mistranslated constructions with light verbs into more tractable, synonymous, constructions. This method reduced the number of incomprehensible translations.

Turcato et al. (2000) used pre-processing techniques to normalise transcribed speech used as input for creating closed captions. They worked primarily with segmentation and proper name recognition, but they believed that the methods used can be useful for a wider range of applications as well.

Aranberri Monasterio (2010) tried different post- and pre-processing methods to improve the quality of translation of English -ing words. The best results came from using rules for automatic source re-writing as pre-processing and from using rules for global search-and-replace as post-processing (Aranberri Monasterio, 2010, p. 228-229, 232-233).

### 2.6.3 Sentence segmentation and text simplification

To divide text into clauses using machine learning methods was the shared task of CoNLL-2001 (Tjong et al., 2001). Six systems participated, using different algorithms for machine learning. The system that achieved the best scores was based on decomposing the problem into combinations of binary decisions and then combining them using a boosting learning algorithm<sup>3</sup> (Carreras and Màrquez, 2001). Carreras and Màrquez' system was the only of the participating systems that used features that contained information about a complete sentence (Tjong et al., 2001, p. 56). Four types of features were used (Carreras and Màrquez, 2001, p. 1):

- Word windows: PoS and relative position for each word in the window
- Chunk windows: tags and relative position for each chunk in the window
- Sentence patterns: punctuation, coordinate conjunctions, the word 'that', relative pronouns and verb phrase-chunks
- Sentence features: the number of occurrences of relevant elements

Orăsan (2000) presented a hybrid method for clause splitting, combining a machine learning algorithm with a set of hand-written rules. A model was trained on a corpus that included PoS-tags and clause boundary markings. First, the trained model was applied to the tagged text, then a set of simple sentence-level rules were applied. A first set of rules was used to remove false positives by identifying verb phrases and making sure there were no clause boundaries inside them. A second set of rules tried to correct false negatives by splitting any clauses still containing two verb phrases. Non-finite clauses were split right before the verb phrase if the previous word was not a conjunction. If the previous word was a conjunction, the split was done before it. If the clause was finite it needed a subject and therefore a search for one was performed in the preceding words and the boundary set before it. If a finite clause ended up without a subject, the nearest appropriate word, or group of words, from the previous clause was used. Orăsan (2000, p. 133) concluded that the rules

---

<sup>3</sup>A boosting algorithm is used to combine many weak features into strong classification rules.

helped improve the results, but that more complex rules would be necessary to achieve better results.

Leffa (1998) created an algorithm for segmenting clauses and identifying them as either noun clauses or adverb clauses. His definition of a clause demanded the presence of a verb phrase but did not require the clause to be finite (Leffa, 1998, p. 6). This definition was suitable for the English-Portuguese machine translation system he developed the algorithm for.

Ejerhed (1999) developed a rule-based algorithm for segmenting PoS-tagged Swedish text into simple, non-nested, clauses. She used eleven rules to define clauses. The rules covered cases with punctuation, conjunction, subjunction and sequences of finite verbs. Evaluation of the algorithm showed 96% correctness when the input was manually tagged and 91% correctness when applied to automatically tagged text. Most errors were due to cases not covered by the rules (Ejerhed, 1999, p. 29).

Lyon and Dickerson (1997) tried to reduce the complexity of parsing by decomposing sentences not into clauses but into three sections: pre-subject, subject and predicate. They suggested that this method can be used as a pre-processing step for parsing, but did not try it in any specific NLP-application.

Kuboň et al. (2007) segmented Czech sentences by applying rules to morphologically analysed text. Since Czech has strict rules for punctuation (Kuboň et al., 2007, p. 368), practically always separating for example two finite verbs with some separator (comma, colon, semicolon or conjunction), they could use this to obtain a small set of rules with high coverage. Their algorithm returned every possible way of segmenting a sentence according to the rules, in the form of segmentation charts. Their rules included commas, coordinating expressions, major delimiters and quotation marks (Kuboň et al., 2007, p. 371).

Kim and Oh (2008) used support vector machines to segment sentences. They worked with PoS-tagged sentences, used a large corpora to create a segmentation model, and experimented with different features to be considered. They evaluated their model according to recall and precision. Large amounts of training data increased performance and made the system benefit from more features.

Several methods have been suggested to solve the task of text simplification. Text simplification can be defined as “any process that reduces the syntactic or lexical complexity of a text while attempting to preserve its meaning and information content” (Siddharthan, 2004, p. 17). Text simplification can be lexical or syntactical: re-writing words into easier ones, or reducing the syntactic complexity of a sentence, often creating several sentences instead. To simplify the syntax, the sentence needs to be analysed. To perform a full parse, however, seems counter-intuitive if the simplification is intended as a pre-processing step to simplify parsing. A more shallow way of parsing is necessary. Chandrasekar et al. (1996) tried two alternatives: chunking and partial parsing with dependency information. The dependency based model was better at handling relative clauses (Chandrasekar et al., 1996, p. 1044).

Siddharthan (2004) syntactically simplified texts by turning appositives, relative clauses and conjoined clauses into new sentences. In the analysis step, noun chunking and clause identification were used, among other methods. The clause identification used mainly punctuation and was rule-based. Simplifica-

tion rules were then used to transform the sentences and regenerate them. Siddharthan (2004, p. 152) suggests that text simplification may be useful as a pre-processing step for machine translation because the simplified sentences might be easier to translate correctly.

## 3 Implementation

We have created a program that matches and segments sentences according to a set of rules. Rules have been developed for three different types of input: tagged and chunked sentences, tagged but not chunked sentences and sentences neither tagged nor chunked.

### 3.1 Segmentation principles

The main task of our work has been to locate where in the sentences segmentation may be executed to possibly increase the translation quality, but at least without decreasing it. To do this, the words and characters that are used to bind clauses together need to be identified. These typically include delimiters, conjunctions and subjunctions.

Ejerhed (1999) reached good results in segmenting Swedish sentences into clauses using rules based on the following criteria (Ejerhed, 1999, p. 28):

- Adding a clause marker after delimiters: . ? ! , - :
- Adding a clause marker before words tagged as subjunctions, adverbs, determiners, pronouns, possessives
- Adding a clause marker before words tagged as conjunctions followed by:
  1. words tagged as subjunction, adverb, determiner, pronoun or possessive
  2. a finite verb
  3. a pronoun, noun, proper name or adverb followed by a finite verb
- Adding a clause marker between verbs, if sequences of finite verbs with 0-2 words between the verbs

We began by using Ejerhed's rules as a base and used Svenska Akademiens grammatik (The Swedish Academy's grammar) (Teleman et al., 1999) to find candidate words. Since these are function words, their numbers are limited. The conjunctions listed are (Teleman et al., 1999, p. 728): *och* ('and'), *samt* ('and'), *eller* ('or'), *men* ('but'), *fast* ('though'), *utan* ('without'), *ty* ('for'), *för* ('for'), *så* ('so'). We also included *dels* ('partly').

Subjunctions are a bit more open than conjunctions, including for example (Teleman et al., 1999, 733-745): *medan* ('while'), *innan* ('before'), *före*

('before'), *efter* ('after'), *sedan* ('then'/'since'/'after'), *att* ('that'), *utan* ('without'), *tills* ('until'), *om* ('if'), *som* ('as'), *huruvilda* ('if'), *än* ('than'), *eftersom* ('as'/'that'/'since'), *då* ('as'/'then'), *samtidigt* ('while'/'at the same time').

Only rules for the clause-binding words actually present in our first set of validation data were considered. These were *att*, *då*, *dels*, *eller*, *för*, *men*, *och*, *om*, *samt*, *sedan* and *som*.

Since the data we have been working with was divided into sentences from start (apart from some of the technical manuals data, see section 5.2), we did not include the major delimiters (. ? !). We did however include some pairwise delimiters. The delimiters evaluated were comma, dash, parenthesis and quotation marks. We believe that it would be safe to also include colon and semi-colon, but since they only occur as the last character of sentences in our validation data we have not been able to evaluate their impact. We also tested segmenting sentences that contained more than one finite verb.

We evaluated candidate rules in an extrinsic way; we did not directly evaluate the way the Swedish sentences were segmented but the quality of the resulting English translations in comparison with the translations of non-segmented sentences. The rules were mainly evaluated using automatic metrics. The automatic metrics used were BLEU, TER and METEOR. To use automatic evaluation during development is fast and efficient, but to better judge fluency and to judge the magnitude of the occurring errors, the translations were also manually examined. This may also compensate for the fact that only one reference translation is used for the automatic evaluation methods. Regarding the magnitude of errors, we considered syntactical errors, such as misses in verb agreement, as more severe problems than errors common also in the non-segmented translations, for example missing commas.

When evaluating rules we also took into consideration the amount of occurrences of the matching patterns. We did not want to customise rules for segmenting our development data, but to find rules that were general enough to match unseen data.

Even though we wanted our set of rules to have high recall we ended up using several more specific rules (see chapter 4). Since long sentences still almost always contain the patterns covered by these smaller sets of rules, this did not have a critical effect on recall. Also, because our goal was not to segment all sentences into clauses but to find safe places to segment the sentences without impeding the translations, we did not want to create segments that could not be analysed correctly by the parser. We tried to avoid segmenting, for example, nested sub-clauses that left the main clause chopped into pieces.

Translation quality may also be affected by the length of the sentences and segments. It is more difficult for the translation system to make complete analyses of long sentences, which may cause detriment to the translation quality. This also applies to very short segments, as they do not necessarily form a complete sentence or clause.

Li et al. (1990) segmented English sentences longer than 40 words and Gerber and Hovy (1998) segmented Japanese sentences if they had a minimum of 20 words. Since we believe these minimum lengths to be language specific, and since these values were calculated 20 and 12 years ago and may not be quite up to date, we have made experiments with different minimum lengths for sentences to find the optimal value for the Swedish sentences we

---

```

input FILE rules
input FILE sentence
input INTEGER min
for all  $s \in \text{sentences}$  do
    if number of words in  $s < min$  then
        add  $s$  to processed
    else
        while number of words in  $s \geq min$  then
            for all  $r \in \text{rules}$  do
                if  $s$  matches left-side of  $r$  then
                    split  $s$ 
                    if all  $\text{segments} \in \text{split } s$  is long enough then
                        recursively split  $\text{segments}$ 
                        for all  $\text{segment} \in \text{split } s$  do
                            add  $\text{segment}$  to processed
                    else
                        break while
                    add  $s$  to processed
            print processed to FILE processed sentences
output FILE processed sentences
output FILE log file

```

---

**Figure 3.1:** Algorithm

have been working with.

We decided to only segment sentences if the resulting segments contained at least three tokens. This is to avoid creating segments consisting of single words.

## 3.2 The program

The program used to segment sentences is implemented in Java. Rules are read from a text file and used to recursively segment sentences. The program needs three arguments: a file containing rules, a file containing (potentially tagged and potentially chunked) sentences to be segmented and an integer that specifies which length sentences should have for them to be segmented. The length given is the minimum number of words. The sentence file should consist of one sentence per line. Sentences are read one by one (actually line by line). The matching is performed using regular expressions. If a part of a sentence matches the left side of a rule, the string is replaced with the right side of the rule and the sentence is split. If the resulting segments are long enough, they are recursively segmented until they are shorter than the integer given or until no match is found. When all sentences have been processed, the resulting segments are written to a file. A log file is also created, containing information on how many times each sentence has been segmented. The log file can be used to rebuild the sentences after translation.

### 3.2.1 Rule file syntax

In the rules, regular expressions are used to match patterns in sentences. Any type of regular expressions compatible with Java can be used. We have used regular expressions of Perl-type (Wall and the Perl 5 development team, 2006) because of their rich syntax.

The rule file should be ordered by falling priority, since the program tries to find the first matching rule. All rules are of one of the three following types:

- Adding a split after a pattern: `X --> X <split>`
- Adding a split before a pattern: `X --> <split> X`
- Adding a split between two patterns: `X Y --> X <split> Y`

In the syntax of the rule file `-->` is used between the expression to be matched and the replacing expression that includes a split-marker. We use parentheses and backreferences (`\1` and `\2`) to mark which part of the expression should appear on which side of the split. For example:

- `(./F.)( och/CCS) --> \1 <split> \2`

would be a rule adding a split between any character tagged as a delimiter and the word *och* ('and') tagged as a conjunction.

## 4 Segmentation rules

All three sets of rules were originally developed using syllabi data, but were also evaluated with text from two other domains (see section 5.2).

The rules for chunked data can be described as a combination of chunk, tag and string matching. String patterns and tag patterns as well as chunk information are used in that set of rules. In the rules for tagged sentences, tag matching and string matching are used. Obviously, only string-matching can be utilised in the rules for non-tagged sentences.

The rules are prioritised in the order they appear below. The order is motivated by the results of the rules for the development data, and by the specificity of the rule. If two rules produce similar results, the rule that covers the longest context is preferred.

### 4.1 Rejected rules

In one way or another, every rule that we have chosen to use has come from a more general one. Our first rules included splitting at every conjunction and at every delimiter. As these rules proved to be moderately successful, we tested all conjunctions and delimiters individually, so as to see which of them would be appropriate to use for splitting.

The results of splitting at every comma were not very good, mainly because one of the most frequent uses for commas is to separate items in listings. Disjoining the words in a listing may have a negative effect on the translation quality because the words before the listing will often be related to the words after the listing. This can to some extent be remedied in the rules for chunked data by stating that it is only suitable to split at commas that do not appear inside a chunk. But, because it is sometimes difficult for the chunker to identify listings, this is not enough to solve the problem. Only splitting at commas not enclosed by identical chunks did not help us come to terms with this problem either.

We encountered this very same problem with conjunctions, to varying degrees. Only the conjunctions *än* ('than'), *innan* ('before') and *medan* ('while') were safe to split at, while *samt* ('as well as', 'and'), *men* ('but'), *dels* ('partly'), *antingen* ('either') and *eftersom* ('because') proved suitable as long as they were not inside a chunk. For all other conjunctions we added the constraint that the conjunction must be preceded by a comma.

Because some very frequently occurring words and patterns that we would have liked to use did not test as well as we had wished, we went on to try combining or adding constraints to unsuccessful rules, just as with the added

constraint to conjunctions of requiring a comma in the previous position.

In an effort to capitalise further on the high frequency of commas (511 occurrences in the development set of 727 sentences) we tried several other patterns of commas followed by some other part of speech. Unfortunately, only one of those patterns proved suitable to add to the rule set: a comma followed by an adverb not used for comparison. The unsuccessful patterns starting with a comma included:

- comma followed by any adverb
- comma followed by a relative determiner
- comma followed by relative pronouns

We also exhausted possible patterns involving conjunctions, including:

- any conjunction followed by an adverb chunk or a preposition chunk or a verb chunk
- any conjunction preceded by a chunk not containing any word in genitive form
- any finite verb followed by as few words as possible (including no words), followed by a conjunction, followed by a noun chunk and a finite verb

From all combinations involving conjunctions we were able to find two successful rules (other than the previously mentioned comma followed by any conjunction):

- any conjunction followed by an infinitive form verb
- any conjunction followed by a infinitive chunk or an infinitive marker and an infinitive form verb

As shown by the candidate rule pattern of any finite verb followed by as few words as possible (including no words), followed by a conjunction, followed by a noun chunk and a finite verb, we tried building rules that would include entire clauses. Unfortunately, the sentences in the data are usually not that linear and very few sentences could be segmented by such a rule. We also found that rules with a longer span did not necessarily prove to be more successful than shorter, less complex, ones.

As for the other delimiters, only semicolon, colon and dash led to positive results. Periods, by definition, will not appear in the middle of sentences unless they are used in abbreviations or numeric expressions (in which case they are treated as part of a token and not as separate tokens). The former also disqualifies the other major delimiters. Splitting at pairwise delimiters led to very poor results, because the words prior to the first delimiter in any pair of delimiters will usually be related to the words after the second one.

## 4.2 Rules for tagged and chunked sentences

There are eight rules for chunked data.

(1) ( ,/FI [^ ]+/C[CS]S) --> \1 <split>

This rule allows splitting of sentences wherein a conjunction follows a comma. The split occurs after the conjunction.

(2) ( ,/FI \[ADVP\\* [^ ]+/R.O. \\*ADVP\]) --> \1 <split>

This rule allows splitting of sentences wherein an adverb not used for comparison follows a comma. The split occurs after the adverb. Adverbs of comparison that occur after a comma are, in our data, usually used in listings of same-PoS words, making it unsuitable to split sentences next to them.

(3) ([;:-]/F[IE]) --> \1 <split>

This rule allows splitting of sentences containing a semicolon, colon or dash. The split occurs after the semicolon, colon or dash. As the tokeniser does not split words containing these tokens there is no risk of the split affecting such words.

(4) ( (samt|men|dels|antingen|eftersom)/C[CS]S \[  
--> <split> \1

This rule allows splitting of sentences containing the following conjunctions: *samt* ('as well as', 'and'), *men* ('but'), *dels* ('partly'), *antingen* ('either'), *eftersom* ('because'). The split occurs before the word. The conjunction must be followed by a chunk for splitting to be allowed. This is to avoid splitting inside same-PoS listings.

(5) ( \[ADVP\\* samt/RGOS \\*ADVP\]) --> <split> \1

This rule allows splitting of sentences containing *samt* ('as well as', 'and') when tagged as an adverb. The split occurs before the word.

(6) ((\] eller|än|innan|medan)/C[CS]S) --> \1 <split>

This rule allows splitting of sentences containing the following conjunctions: *eller* ('or'), *än* ('than'), *innan* ('before'), *medan* ('while'). The split occurs after the word. *Eller* ('or') must occur after a chunk. This is to avoid splitting inside same-PoS listings.

(7) ( [^ ]+/C[CS]S \[VC\\* [^ ]+/V@N[^\*]+\\*VC\]) --> <split> \1

(8) ( [^ ]+/C[CS]S \[INFP\\* [^ ]+\\*INFP\]) --> \1 <split>

These rules allow splitting of sentences wherein an infinitive verb phrase follows a conjunction. The infinitive verb phrase may consist of either a verb in infinitive form or of an infinitive marker and a verb in infinitive form. One rule is for verb chunks only containing a verb in infinitive form, and the other is for infinitive chunks (these always contain an infinitive marker and a verb in infinitive form). In the first case the split occurs before the conjunction, and in the second case the split occurs after the infinitive chunk.

## 4.3 Rules for tagged sentences

This set of seven rules is basically the same as the one for chunked data, save for some restrictions not being applicable without the chunk information.

(1) ( ,/FI [^ ]+/C[CS]S) --> \1 <split>

This rule is the same as rule number one in the previous set.

(2) ( ,/FI [^ ]+/R.O.) --> \1 <split>

This rule allows splitting of sentences wherein an adverb not used for comparison follows a comma.

(3) ([;:-]/F[IE]) --> \1 <split>

This rule is the same as rule number three in the previous set.

(4) ( (samt|men|dels|antingen|eftersom)/C[CS]S)  
--> <split> \1

This rule allows splitting of sentences containing the following conjunctions: *samt* ('as well as', 'and'), *men* ('but'), *dels* ('partly'), *antingen* ('either'), *eftersom* ('because'). The split occurs before the word.

(5) ( samt/RGOS) --> <split> \1

This rule allows splitting of sentences containing *samt* ('as well as', 'and') when tagged as an adverb. The split occurs before the word.

(6) ( (eller|än|innan|medan)/C[CS]S) --> \1 <split>

This rule allows splitting of sentences containing the following conjunctions: *eller* ('or'), *än* ('than'), *innan* ('before'), *medan* ('while'). The split occurs after the word.

(7) ( [^ ]+/C[CS]S( att/CIS)? [^ ]+/V@N[^ ]+) --> <split> \1

This rule allows splitting of sentences wherein an infinitive verb follows a conjunction. The split occurs at the position before the conjunction.

## 4.4 Rules for non-tagged sentences

The five rules for non-tagged sentences are a bit different than the other two sets since it is not possible to make certain restrictions without the PoS-tags. The rule(s) in the tagged and chunked data that allows splitting at infinite form verbs, possibly preceded by an infinitive marker, is not possible to recreate sensibly without PoS information. Therefore, this rule is not included in the rule set for non-tagged data.

(1) ( , ( antingen|både|dels|dock|eftersom|eller|emedan|fast|för|  
huruvida|innan|medan|men|och|om|samt|som|såsom|såväl|tills|  
utan) ) --> \1 <split>

This rule allows splitting of sentences wherein a conjunction follows a comma. The words used are the conjunctions found in the tagged validation syllabi data. The split occurs after the conjunction.

(2) ( , ( hur|inte|respektive|ej|dessutom|t.ex.|till exempel|icke|  
tillsammans|sedan|huvudsakligen|varav|ovan|ca|cirka) )  
--> \1 <split>

This rule allows splitting of sentences wherein an adverb follows a comma. The adverbs used are the ones that most frequently appear as non-comparison adverbs in the tagged validation syllabi data.

(3) ( [;:-] ) --> \1 <split>

This rule allows splitting of sentences containing a semicolon, colon or dash. The split occurs after the semicolon, colon or dash.

(4) ( ( samt|men|dels|antingen|eftersom) ) --> <split> \1

This rule allows for splitting sentences containing the following conjunctions: *samt* ('as well as', 'and'), *men* ('but'), *dels* ('partly'), *antingen* ('either'), *eftersom* ('because'). The split occurs before the word.

(5) ( ( eller|än|innan|medan) ) --> \1 <split>

This rule allows splitting of sentences containing the following conjunctions: *eller* ('or'), *än* ('than'), *innan* ('before'), *medan* ('while'). The split occurs after the word.

## 5 Evaluation

In this chapter we will account for the performed methods of evaluation and the data utilised. We will also present the results of the evaluation.

### 5.1 Method

We have used extrinsic evaluation, meaning that we have evaluated the result of the translated data. To use intrinsic evaluation, and evaluate specifically the precision and recall of the segmenter, would not be particularly useful for our purposes. Our goal is to improve the efficiency and quality of machine translation, not to in a linguistically correct way segment sentences.

The most essential evaluation factors for this task are processing time and translation quality. The presented translation times are the actual processing times for the translation system, independent of CPU usage and server workload, and not including the added time for segmentation, tagging and chunking performed prior to translation.

We compared the quality of translations of segmented sentences with those of non-segmented sentences. Translations were automatically evaluated using three different metrics: BLEU (see section 2.5.3), TER (see section 2.5.5) and METEOR (see section 2.5.4). We used the default settings for all metrics.

The possibility of errors being caused during tagging and chunking needs to be taken into consideration. The Swedish language model for HunPos that we have used is trained on the SUC-corpus (Ejerhed and Källgren, 1997). The model has been evaluated on a part of the corpus by Megyesi (2009) with an accuracy of 95.9%. The result of the chunking is of course dependent on the result of the tagging.

The translations were performed using domain-specific lexicons for the syllabi and the technical manuals. There is currently no lexicon available for the patent texts, these were translated using the same lexicon as the technical manuals. This most certainly had an impact on the translation quality for these texts.

### 5.2 Data

The first sets of validation and test data consist of syllabi sentences, with corresponding translations, that have been created by the translation system and manually edited by the users. We received the sentences in alphabetical order of the Swedish sentences first letter and divided the sentences into two data sets. To get the sets varied, every other sentence was moved to the validation

set and the rest to the test set. This was achieved using shell scripts. During development we also used a smaller development set consisting of 727 sentences from the validation set: every tenth of the sentences with a length of at least four words. This smaller set was used to obtain reasonable translation times when testing a large number of variations of rules.

The syllabi sentences are of rather varying lengths. The texts concern education, course requirements and specifically courses within the medical field. These are two example sentences from the test set:

**Swedish sentence:** *Att skriva vetenskaplig rapport, 3,5hp*

**Reference translation:** *How to write a scientific report, 3.5 HE credits*

**Swedish sentence:** *Dessutom ger kursen central kunskap om psykopatologi hos barn och vuxna, psykiatrisk diagnostisering, samt övergripande kunskap om olika teoretiska perspektiv (kognitivt, inlärningspsykologiskt, psykodynamiskt) och evidensen av interventioner baserade på dessa.*

**Reference translation:** *Furthermore, the course provides central knowledge of psychopathology in children and adults, psychiatric diagnoses and general knowledge of various theoretical perspectives (cognitive, learning psychological, psychodynamic) and the evidence of interventions based on these.*

For the patent applications and technical manuals we used considerably smaller validation sets (see table 5.1). The reason for this is that translation times for large data sets are extensive. We did not use these sets to develop the segmenter and the segmentation rules, but only to make sure that everything worked as expected before evaluating the system on the test data sets for these two domains.

The patent application data derives from a translation memory used by human translators and the translations are manually made. Sentences in these texts are long and complex. There are also a lot of formulas and other structures that do not form grammatical clauses. To be able to process the data within reasonable time, we used a considerably smaller set of test sentences. Two example sentences are:

**Swedish sentence:** *En "primer" är en oligonukleotid som är tillräckligt komplementär till en mall så att den hybridiseras (genom vätebindning eller hybridisering under hybridiserande betingelser, t.ex. stringenta betingelser) med mallen och ger ett primer/mall-komplex lämpligt för initiering av syntes med ett DNA-polymeras, såsom omvänt transkriptas, och som förlängs genom addition av kovalent bundna baser länkade till dess 3'-ände som är komplementära till mallen.*

**Reference translation:** *A primer is an oligonucleotide that is sufficiently complementary to a template so that it hybridizes (by hydrogen bonding or hybridization under hybridizing conditions, e.g., stringent conditions) with the template to give a primer/template complex suitable for initiation of synthesis by a DNA polymerase, such as a reverse transcriptase, and which is extended by the addition of covalently bonded bases linked to its 3 end that are complementary to the template.*

**Swedish sentence:** *I synnerhet tillhandahåller systemet och metoden enligt föreliggande uppfinning ett händelsestyrt datorprogram för omdirigering (omdiriger-*

ingsprogram"), som, då en särskild användardefinierad händelse har inträffat, omdirigerar användarvalda dataposter från värdsystemet till användarens mobila datakommunikationsutrustning.

**Reference translation:** *In particular, the system and method of the present invention provide an event-driven redirection computer program (redirector program) operating at the host system, which, upon sensing a particular user-defined event has occurred, redirects user-selected data items from the host system to the users mobile data communication device.*

The third data sets were technical manuals from a company manufacturing trucks. The manuals are used to give information on how to, for example, repair a truck. The data comes from a translation memory of manually made translations. The sentences in these data sets are generally shorter, but there are a lot of incomplete sentences, abbreviations and lists. Also, a large number of units in the translation memory actually consist of two or more sentences, so the average sentence length is in reality even shorter. The translations are not always one-to-one sentence-wise. Examples of sentence units in the test set are:

**Swedish sentence:** *Läs av värdena på indikatorklockan i båda riktningar. Summan av dem ska vara inom 0,15-0,25mm.*

**Reference translation:** *Take readings from the dial gauge in both directions. The total of these should be within the range 0.15-0.25mm.*

**Swedish sentence:** *Övrigt: Fordon med DC9-, DC11- eller DC12-motor.*

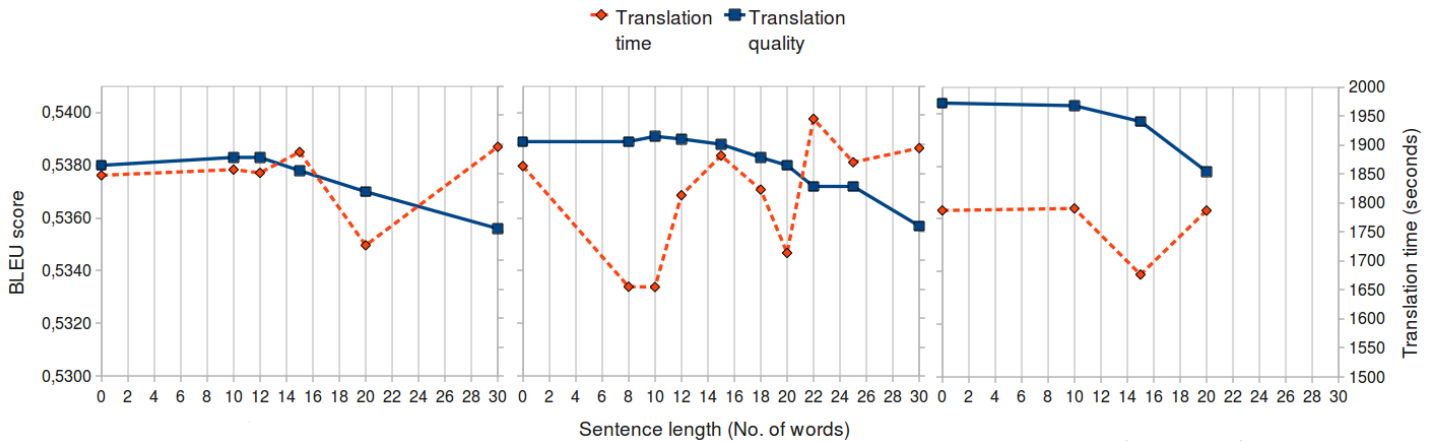
**Reference translation:** *Others: Vehicles with a DC9, DC11 or DC12 engine.*

**Table 5.1:** Data properties

Validation data	Sentences	Words	Average sentence length (No. of words)	Longest sentence (No. of words)
Syllabi	8326	95806	11.5	243
Patent applications	312	7361	23.6	110
Technical manuals	511	6939	13.6	94
Test data	Sentences	Words	Average sentence length (No. of words)	Longest sentence (No. of words)
Syllabi	8327	94848	11.4	134
Patent applications	1173	26206	22.3	302
Technical manuals	11140	77684	7.0	106

### 5.3 Minimum sentence lengths

We used the development set for syllabi data to test different minimum lengths for sentences to be segmented. The results can be seen in figure 5.1.



**Figure 5.1:** Syllabi development set results for different minimum sentence lengths. From left to right: chunked data, tagged data, non-tagged data

**Table 5.2:** Translation quality for the different data sets and rule sets

	Non-segmented data			Non-tagged data		
	BLEU	METEOR	TER	BLEU	METEOR	TER
Syllabi	0.5431	0.8068	0.2732	0.5404	0.8056	0.2749
Patent applications	0.2354	0.5834	0.5563	0.2381	0.5864	0.5526
Technical manuals	0.3527	0.7155	0.4345	0.3716	0.7257	0.4145
	Tagged data			Chunked data		
	BLEU	METEOR	TER	BLEU	METEOR	TER
Syllabi	0.5375	0.8038	0.2788	0.5376	0.8038	0.2787
Patent applications	0.2382	0.5861	0.5572	0.2379	0.5859	0.5572
Technical manuals	0.3688	0.7247	0.4180	0.3688	0.7247	0.4180

We wanted to achieve high BLEU scores and low translation times. We also wished to find a single value that gave generally good results and could be used for all rule sets. There are no distinct patterns in the curves, probably due to the relatively small data set used. However, by taking into consideration the quality and the time values for each of the rule sets we decided to segment sentences that had more than 10 words.

## 5.4 Results

In this section we will show the translation quality and processing time for the different rule sets and data sets. As seen in table 5.2, the three different quality metrics consistently show high correlation. Because of this, and for the sake of clarity, we will use BLEU as the only comparative measure of quality for the following result comparisons.

Table 5.4 shows the BLEU scores and translation times for the syllabi test data. The scores are overall a bit lower than for the non-segmented data, but the processing times are significantly reduced. The rules for non-tagged data

**Table 5.3:** Number of sentences segmented and total number of segments (including non-segmented sentences) produced by the different rule sets, for the different domains

		Non-tagged data	Tagged data	Chunked data
<b>Syllabi test data</b> 8327 sentences	No. of segmented sentences	1103	1723	1669
	% segmented sentences	13.25	20.69	20.04
	No. of segments	9539	10519	10450
<b>Patent application test data</b> 1173 sentences	No. of segmented sentences	319	414	387
	% segmented sentences	27.20	35.29	32.99
	No. of segments	1600	1786	1735
<b>Technical manuals test data</b> 11140 sentences	No. of segmented sentences	489	751	714
	% segmented sentences	4.40	6.74	6.41
	No. of segments	11704	12025	11965

**Table 5.4:** Syllabi data results for segmented sentences and non-segmented sentences

Translation times in seconds			
	segmented data	non-segmented data	difference (%)
Non-tagged data	16957.75	21344.61	-20.55
Tagged data	16041.07	"	-24.85
Chunked data	16603.26	"	-22.21
BLEU scores for all sentences			
	segmented data	non-segmented data	difference
Non-tagged data	0.5404	0.5431	-0.0027
Tagged data	0.5375	"	-0.0056
Chunked data	0.5376	"	-0.0055
BLEU scores for only the segmented sentences			
	segmented data	non-segmented data	difference
Non-tagged data	0.4999	0.5097	-0.0098
Tagged data	0.5163	0.5294	-0.0131
Chunked data	0.5193	0.5324	-0.0131

achieve the highest score, but also the longest translation times. This set of rules produces the lowest number of segments (see table 5.3). The results for tagged and chunked data are rather similar, and they do perform almost the same amount of segmentations on the test data. Table 5.4 also shows that the BLEU scores are generally lower for the segmented sentences than for the entire data set.

In table 5.5, results for the patent application data are presented. The BLEU scores are generally low, and are not noticeably improved by the segmentation. Translation times are, however, clearly reduced. The relation between the different rule sets is similar to that for the syllabi data. As seen in table 5.3 this domain has the highest percentage of segmented sentences.

The BLEU scores and the translation times for the technical manuals are presented in table 5.6. These sentences are generally short, and a lot fewer of

**Table 5.5:** Patent application data results for segmented sentences and non-segmented sentences

<b>Translation times in seconds</b>			
	segmented data	non-segmented data	difference (%)
Non-tagged data	18099.62	22693.01	-20.24
Tagged data	15943.76	"	-29.74
Chunked data	16236.31	"	-28.45
<b>BLEU scores for all sentences</b>			
	segmented data	non-segmented data	difference
Non-tagged data	0.2381	0.2354	0.0027
Tagged data	0.2382	"	0.0028
Chunked data	0.2379	"	0.0025
<b>BLEU scores for only the segmented sentences</b>			
	segmented data	non-segmented data	difference
Non-tagged data	0.2282	0.2244	0.0038
Tagged data	0.2408	0.2398	0.0010
Chunked data	0.2464	0.2458	0.0006

**Table 5.6:** Technical manuals data results for segmented sentences and non-segmented sentences

<b>Translation times in seconds</b>			
	segmented data	non-segmented data	difference (%)
Non-tagged data	16531.82	17946.48	-7.88
Tagged data	15769.88	"	-12.13
Chunked data	16069.50	"	-10.46
<b>BLEU scores for all sentences</b>			
	segmented data	non-segmented data	difference
Non-tagged data	0.3716	0.3527	0.0189
Tagged data	0.3688	"	0.0161
Chunked data	0.3688	"	0.0161
<b>BLEU scores for only the segmented sentences</b>			
	segmented data	non-segmented data	difference
Non-tagged data	0.3227	0.2957	0.0270
Tagged data	0.3332	0.3092	0.0240
Chunked data	0.3334	0.3087	0.0247

them are segmented than in the other two domains (see table 5.3). Translation times are still reduced and BLEU scores are improved. The segmented sentences overall achieve improved BLEU scores. However, scores are a bit misleading because of differences in the handling of punctuation marks that, in this domain, favour the segmented data to a greater extent (see chapter 6 for further remarks on this).

## 5.4.1 Analysis of example sentences

In this section we will show some examples of sentences that have been translated differently in their segmented form than in their non-segmented form. Some of the sentences show where the segmentation has helped the parser, resulting in a better translation, and others exemplify problems that arise because of the segmentation.

**Syllabi sentence:** *Datasökning, litteraturstudier, och rapportskrivning är viktiga delar i kursen.*

**Reference translation:** *Data search, literature studies and report writing are important parts of the course.*

**MT output without segmentation:** *Data search, literature studies and report writing are important parts in the course.*

**MT output with segmentation:** *Data search, literature studies, and report writing is important parts in the course.*

The segmented sentence has been split after *och* ('and'), matching the pattern ( ,/FI [^ ]+/C[CS]S). This has led to 'is' only referring to 'report writing', instead of 'data search, literature studies and report writing'. Because using a comma before *och* ('and') in a listing is not very common, this error does not occur frequently.

**Syllabi sentence:** *Dessutom att kritiskt analysera och bedöma annan students examensarbete.*

**Reference translation:** *Furthermore, to analyse and assess another student's degree project critically.*

**MT output without segmentation:** *Furthermore to analyse and assess other student's degree project critically.*

**MT output with segmentation:** *Furthermore to analyse critically and assess other student's degree project.*

The segmented sentence has been split before *och* ('and'), as it matches the pattern ( [^ ]+/C[CS]S \[VC\\* [^ ]+/V@N[^\*]+\\*VC\]). Because of the split, 'critically' could not be relocated to the end of the sentence, but only to the end of the segment. This has led to 'critically' only applying to 'analyse' instead of 'analyse and assess other student's degree project'. This is a more serious problem with the segmentation, but does not always make the translations worse.

**Syllabi sentence:** *Dessutom förväntas deltagarna förbereda ett försöksprotokoll och analysera en vetenskaplig artikel.*

**Reference translation:** *Furthermore, the participants are expected to prepare an experimental protocol and analyse a scientific article.*

**MT output without segmentation:** *Furthermore, the participants are expected prepare an experimental record and analyse a scientific article.*

**MT output with segmentation:** *Furthermore, the participants are expected to prepare an experimental record and analyse a scientific article.*

The segmented sentence has been split before *och* ('and'), matching the pattern ( [^ ]+/C[CS]S \[VC\\* [^ ]+/V@N[^\*]+\\*VC\]). The split seems to have enabled the translation system to make a complete analysis, leading to a correct translation.

**Syllabi sentence:** *Examinator kan med omedelbar verkan avbryta en students verksamhetsförlagda utbildning (VFU) eller motsvarande om studenten visar sådana allvarliga brister i kunskaper, färdigheter eller förhållningssätt att patientsäkerheten eller patienternas förtroende för sjukvården riskeras.*

**Reference translation:** *The examiner may with immediate effect interrupt a student's placement (VFU), or the equivalent, if the student demonstrates such serious deficiencies in knowledge, skills or attitudes that patient safety or patient confidence in healthcare is at risk.*

**MT output without segmentation:** *Examiner can with immediate impact interrupt a students placement (VFU) or equivalent if the student shows such serious deficiencies in knowledge, skills or attitude that the patient security or the patients' trust for the healthcare are jeopardised.*

**MT output with segmentation:** *Examiner can with immediate impact interrupt a student's placement (VFU) or equivalent if the student shows such serious deficiencies in knowledge skills or attitude that the patient security or the patients' trust for the healthcare is jeopardised.*

The segmented sentence has been split after both occurrences of the word *eller* ('or'), as it matches ((\ eller| än| innan| medan)/C[CS]S). The split seems to have helped the translation system make a better analysis, leading to a correct verb tense. However, in the segmented sentence a comma has been dropped.

**Patent application sentence:** *Vid drift av värmeelementen i band 52, kommer ökningen eller minskningen av effekten som tillförs värmeelementen att utvidga respektive komprimera band 52 i en radiell riktning så att borsttätningsegmenten 54 förskjuts i spår 56, varigenom spaltbredden mellan borstspetsarna och ytan av den roterbara komponenten kontrolleras aktivt.*

**Reference translation:** *Upon actuation of the heating elements within band 52, the increase or decrease of power supplied to the heating elements will respectively expand or contract band 52 in a radial direction to displace brush seat segments 54 in groove 56, thereby positively adjusting the clearance between the bristle tips and the surface of the rotatable component.*

**MT output without segmentation:** *At operation of the heater elements in band 52, the increase or the reduction of the output that is added come the heater elements to expand or compress band 52 in a radial direction so that borsttätningsegmenten 54 are displaced in groove 56 whereby the gap width between the borstspetsarna and the surface of the roterbara the component be checked actively.*

**MT output with segmentation:** *At operation of the heater elements in band 52, the increase comes or the reduction of the output that is added the heater elements to expand or compress band 52 in a radial direction so that borsttätningsegmenten 54 are displaced in groove 56 whereby the gap width between the borstspetsarna and the surface of the roterbara the component be checked actively.*

This sentence has been segmented after *eller* ('or'), as it matches the pattern ((\ eller| än| innan| medan)/C[CS]S), and after *varigenom* ('whereby'), matching the pattern ( ,/FI \[ADVP\\* [\^ ]+/R.O. \\*ADVP\]). Because of the first split, the transpositioning of *kommer* ('will', 'come') has been blocked. Had it been easier to identify listings correctly, we would probably be able to avoid this split. However, in this context *kommer* should have been translated to 'will', which is not the case in either translation. There are several errors in both translations, but other than the handling of *kommer* the translations are identical.

**Patent application sentence:** *Det partiella vätgastrycket under reduktionen sträcker sig från 1,01 till 101,3 bar (1 till 100 atmosfärer), företrädesvis från 1,01 till 40,52 bar (1 till 40 atmosfärer), och gasvolymshastigheterna per timme är från 100 V/Hr/V till 40.000 V/Hr/V, företrädesvis från 1.000 V/Hr/V till 20.000 V/Hr/V, uttryckta som standardvolym per gasen eller de respektive gasblandningarna (25 grader C, 1,01 bar (1 atm)) per timme per volym katalysator.*

**Reference translation:** *Hydrogen partial pressure during the reduction would range from 1.01 to 101.3 bar (1 to 100 atmospheres), preferably from 1.01 to 40.52 bar (1 to 40 atmospheres), and the gas hourly space velocities would be from 100 V/Hr/V to 40,000 V/Hr/V, preferably from 1,000 V/Hr/V to 20,000 V/Hr/V, expressed as standard volumes of the gas or gas mixtures (25 degrees C; 1.01 bar (1 atm)) per hour per volume of catalyst, respectively.*

**MT output without segmentation:** *The partial hydrogen gas pressure under the reduction extends from 1,01 to 101,3 supported (1 to 100 atmospheres), preferably from 1,01 to 40,52 supported (1 to 40 atmospheres), and the gas volume speeds per hour is from 100 V/Hr/V to 40.000 V/Hr/V, preferably from 1.000 V/Hr/V to 20.000 V/Hr/V, expressed as standard volumes for the gas or the respective gas mixtures (25 degrees C, 1,01 supported (1 atm)) per hour per volume catalytic converter.*

**MT output with segmentation:** *The partial hydrogen gas pressure under the reduction extends from 1,01 to 101,3 supported (1 to 100 atmospheres), preferably from 1,01 to 40,52 supported (1 to 40 atmospheres), and the gas volume speeds per hour are from 100 V/Hr/V to 40.000 V/Hr/V, preferably from 1.000 V/Hr/V to 20.000 V/Hr/V, expressed as standard volumes for the gas or the respective gas mixtures (25 degrees C, 1,01 supported (1 atm)) per hour per volume catalytic converter.*

The segmented sentence has been split after both occurrences of *företrädesvis* ('preferably') and after *och* ('and') and *eller* ('or'), splitting the sentence into five segments. The patterns matched were ( ,/FI \[ADVP\\* [\^ ]+/R.O. \\*ADVP\]), ( ,/FI [\^ ]+/C[CS]S) and ((\ eller| än| innan| medan)/C[CS]S). Although the sentence is very long and, because of that, difficult to translate, none of the individual segments in this sentence are very complex. Because of the multiple splitting, the segmented translation does not contain any grammatical errors. The non-segmented sentence has translated *är* incorrectly. The translation quality of this sentence is reflective of the entire domain. Most errors stem from the lack of a domain-specific dictionary.

**Technical manual sentence:** *Den nya kylvätskepumpen har ett lager med större kapacitet, och tätningen och pumphuset har modifierats för att minimera risken för kylvätskeläckage på grund av luft i kylvätskesystemet.*

**Reference translation:** *The new coolant pump has a bearing of greater capacity while the sealing arrangements and pump housing have been modified to minimize the risk of leakage due to air in the coolant system.*

**MT output without segmentation:** *The new coolant pump has a bearing with larger capacity and the seal and the pump housing has been modified to minimise the risk for coolant leaks due to air in the cooling system.*

**MT output with segmentation:** *The new coolant pump has a bearing with larger capacity, and the seal and the pump housing have been modified to minimise the risk for coolant leaks due to air in the cooling system.*

The segmented sentence has been split after the first occurrence of *och* ('and'), as it matches the pattern ( ,/FI [^ ]+/C[CS]S). In this case, the segmentation has made it more clear that 'the seal and the pump housing' is the subject of 'have'. In the non-segmented sentence it is likely that only 'the pump housing' has been chosen as the subject, giving *har* ('have'/'has') the incorrect tense. That the segmentation has led to a more correct translation of this sentence is entirely due to the placement of the comma.

**Technical manual sentence:** *Dessa anslutningar finns tillgängliga men används inte av Scania:*

**Reference translation:** *These connections are available but not used by Scania:*

**MT output without segmentation:** *These connections are available but are used not by Scania:*

**MT output with segmentation:** *These connections are available but is used not of Scania:*

The segmented sentence has been split before *men* ('but'), matching the pattern ( [^ ]+/C[CS]S \[VC\\* [^ ]+/V@N[^\*]+\\*VC\]). In the segmented sentence, the wrong tense has been chosen for *används* ('are used'). Because of the ellipsis, there is no subject for the verb phrase to refer to after the segmentation. The lack of a subject noun phrase may also be the cause of the incorrect translation of *av* ('by'/'of') in the segmented sentence.

## 6 Discussion

The rules for chunked data and tagged data produce very similar results. The rules for non-tagged data produce similar results in terms of quality, but do not reduce the processing time as much as the two other rule sets. Seeing as the rules for both tagged data and non-tagged data have been adapted from the rules for chunked data, this is no surprise. That the rules for non-tagged data have a lower impact on processing time is not unexpected either; some of the rules for chunked and tagged data necessarily rely on PoS information and consequently could not be adapted to fit the non-tagged data, resulting in fewer segmented sentences.

Errors due to tagging and chunking of course exist, even though the tools used have high accuracy. The chunking step is also affected by errors in tagging, so the results for the chunked data are probably most affected by errors prior to segmentation.

Even if the results for non-tagged data would be similar in quality to the tagged data, it would make no sense for us not to use the rules for tagged data. This is because the Convertus translator tags the data before translating it, meaning that the PoS-tags are available without adding any extra time to the process. The rules for chunked data, however, would have to lead to significantly better results for it to be worth the added time caused by sending the data through the SPARK chunker. The chunking takes approximately one quarter of a second per sentence in the syllabi development set and the validation sets for patent applications and technical manuals. For longer texts, this adds up to a noticeable delay.

The segmentation does not add much time to the process. Even when segmenting data consisting of thousands of sentences, this step is performed in a few seconds.

The translation quality differs significantly between the domains. The relatively high quality of the syllabi translations can be explained by the strict one to one relationship between sentence and entry in the data and the quality of the domain-specific dictionary. The translations of the patent data are of the lowest quality of the three domains. Part of the explanation for this is that the sentences in the patent data are generally longer and more complex, but even more critical may be the lack of a dictionary specifically developed for its domain. Instead of having a patent-specific dictionary, the patent data is translated using settings adapted for technical manuals. This is because the patent translation system is not yet in use. Although the sentences are relatively long, there is a strict one to one relationship between sentence and entry in the data.

The syllabi data and, to a slightly lesser extent, the patent data mainly contain actual sentences. This is not necessarily the case with the technical

manuals, where the relationship between sentence and entry is not always one to one. Many entries actually contain several sentences and many consist only of formulas. The formulas do not present a problem, as they often do not need to be translated. The loose grouping of sentences, however, do. If several sentences are grouped into one entry, we may end up with segments containing the end of one sentence and the beginning of another, which would very likely complicate the segment analysis. Adding a rule that allowed splitting at all major delimiters would deal with that problem, but create another; it is not uncommon that sentences contain abbreviations where the period is separated from the word with a space. The period will then be treated as a separate token, making it available for splitting.

Not at all unexpectedly, segmenting the technical manuals did not cut as much time off the translation process as segmenting the syllabi data did. This can largely be attributed to the sentences in the technical manuals being shorter, on average, than the sentences in the syllabi data. In fact, the average sentence length for the technical manuals is shorter than the minimum sentence length allowed for splitting. The shorter sentence lengths also means that the sentences are less likely to contain any of the rule patterns. And, as figure 5.1 shows, segmenting sentences shorter than the minimum length we decided on does not improve translation quality or translation time.

We believe the small differences in results between different sentence lengths to be owing to the fact that the rules are quite specific. The patterns used primarily occur in longer sentences.

There is one notable difference between the evaluation results of the segmented and the non-segmented data. While the non-segmented translations remain unchanged between translation and evaluation, the segmented translations are cleaned up as a part of reattaching the segments into sentences. This is to make sure that there are no extra spaces between tokens, but it does play a small part in the evaluation. How small of a part differs between the domains. The syllabi reference data is not consistent in the handling of spaces next to delimiters, so neither the segmented nor the non-segmented data seems consistently favoured. Below are two examples of this divergence from the syllabi data. The first one favours the non-segmented output and the second one the segmented.

**Reference translation:** *1 . Written examination*

**MT output without segmentation:** *1 . Written examination*

**MT output with segmentation:** *1. Written examination*

**Reference translation:** *1. Describe theoretical behavioural models of change and learning.*

**MT output without segmentation:** *1 . Describe theoretical behavioural models about change and learning.*

**MT output with segmentation:** *1. Describe theoretical behavioural models about change and learning.*

As mentioned in section 5.4, this has a slightly larger impact on the technical manuals, where the segmented translations are favoured more often. The segmented patent data also achieves some increase in scores because of this,

but to a lesser extent. This problem could be diminished by using alternative reference translations.

The reference translations for the patent applications and the technical manuals are manually created translations. However, the reference translations for the syllabi data consist of machine translation output that has been manually edited by the users. This might lead to some bias towards the non-segmented version for translations where the segmented sentence has ended up with a possible variation, for example in word order. This is an example from the syllabi test set:

**Syllabi sentence:** *Den teoretiska delen består av deltagande i institutionens seminarier, olika forskningsrelaterade aktiviteter i forskningsgruppen samt studier av kurslitteratur.*

**Reference translation:** *The theoretical part consists of participation in the seminar series, different research-related activities in the research group and studies of the reading list of the department.*

**MT output without segmentation:** *The theoretical part consists of participation in the seminar series, different research-related activities in the research group and studies of reading list of the department.*

**MT output with segmentation:** *The theoretical part consists of participation in the seminar series of the department, different research-related activities in the research group and studies of reading list.*

The ambiguity in this sentence is where to attach the translation of the genitive *institutionens* ('of the department'). When reading the Swedish sentence we believe that the segmented MT output actually makes the correct choice. The lack of alternative reference translations for evaluation is a problem.

Prepositional phrases that in the source text appear early in a sentence are often relocated to the tail of the translated sentence. Segmenting a sentence may block this, as the prepositional phrase will not necessarily be in the same segment as the end of the sentence after the segmentation. This also applies to sentence adverbs. However, this does not always impair the translation quality; in some cases it even leads to more accurate results. The example sentence above also happens to illustrate this.

The translation system's handling of commas is not consistent. Very often we have found that the only difference between the result of a segmented sentence and the result of the corresponding non-segmented sentence is that one of the translations has dropped a comma; which of the sentences this happens to seems random. The reason for this is most likely found in Multra's sentence generation.

We have not been able to examine whether or not the segmentation increases the number of hits in the translation memory. The syllabi data used actually is the translation memory used by the company, thus every sentence would get an exact hit. There are currently no up-and-running translation systems with translation memories for the other two domains.

## 7 Conclusion

Our objective was to reduce translation times in a hybrid machine translation system by segmenting the sentences before translation. We created a rule-based sentence segmenter, and managed to reduce translation times for all three domains the system was tested on. The longer the sentences in the domain were, the more segments were created and the reduction in translation time became greater.

We had also hoped that the segmentation would improve translation quality by reducing the complexity of the input to the parser. The quality, however, was not improved. Some of the rules used might be optimised, added or removed which could improve quality, but risk to also decrease the number of segments created, and therefore not reduce translation times as much. Some of the errors caused by segmentation (as exemplified in section 5.4.1) might be corrected by post-processing rules, but we believe that in order to achieve any significant quality improvements while retaining low translation times, it is probably necessary to go about this task another way.

Future work in this area could include a more statistical approach towards learning segmentation rules and which rule to prefer in which context. A feature-based classifier, with access to sufficient amounts of segmented training data, would probably be able to perform well at this task.

It might also be beneficial to not treat each part of a segmented sentence as a completely new possible match for segmentation, but to instead perform different segmentations depending on the previous ones performed on that sentence. This would call for a large set of specific rules and methods for how to keep track of the dependencies and priorities.

Another possible improvement would be to add rule-specific requirements for the segments, for example demanding a word of a specific PoS, or a specific type of phrase or chunk, to appear. Rule-specific requirements for segment lengths could also be added.

To use a more text-simplification based approach for this task would probably make it possible to identify a larger number of segments. By making sure every segment creates a complete sentence, problems with, for example, verb agreement would potentially disappear. It could also be beneficial to use a method that makes it possible to keep main clauses intact while detaching nested sub-clauses. This would increase the number of segments and simplify the text. However, the task of re-attaching the clauses after translation would create a need for good post-processing rules.

## Bibliography

- ALPAC. *Languages and Machines: Computers in Translation and Linguistics. A report by the Automatic Language Processing Advisory Committee, Division of Behavioral Sciences, National Academy of Sciences, National Research Council.* National Academy of Sciences, National Research Council, Washington, D.C., 1966. Publication 1416.
- Nora Aranberri Monasterio. *-ing Words in RBMT: Multilingual Evaluation and Exploration of Pre- and Post-processing Solutions.* PhD thesis, School of Applied Language and Intercultural Studies, Dublin City University, 2010.
- Doug Arnold. Why translation is difficult for computers. In Harold Somers, editor, *Computers and Translation: A Translator's Guide*, pages 119–142. John Benjamins Publishing Company, Amsterdam, the Netherlands, 2003.
- John Aycock. Compiling Little Languages in Python. In *Proceedings of the 7th International Python Conference*, 1998.
- Bogdan Babych, Anthony Hartley, and Serge Sharoff. Evaluation-guided pre-editing of source text: improving MT-tractability of light verb constructions. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, pages 36–43, Barcelona, Spain, 2009.
- Satanjeev Banerjee and Alon Lavie. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43th Annual Meeting of the Association of Computational Linguistics (ACL-2005)*, pages 249–256, Ann Arbor, Michigan, June 2005.
- Björn Beskow. *Generation in Multra.* Uppsala University. Department of Linguistics, 1997.
- Björn Beskow. *Unification-Based Transfer in Machine Translation. RUUL 24.* Uppsala University. Department of Linguistics, 1993.
- Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. *Bracketing Guidelines for Treebank II Style Penn Treebank Project.* Linguistic Data Consortium, 1995.
- Torsten Brants. TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference*, Seattle, Washington, 2000.
- Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21, 1995.

- Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006.
- Xavier Carreras and Lluís Màrquez. Boosting trees for clause splitting. In *Proceedings of the Fifth Workshop on Computational Language Learning (CoNLL-2001)*, pages 1–3, Toulouse, France, July 2001.
- Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. Motivations and Methods for Text Simplification. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, pages 1041–1044, Copenhagen, Denmark, August 1996.
- Michael Collins, Philipp Koehn, and Ivona Kučerová. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL-2005)*, pages 531–540, Ann Arbor, Michigan, June 2005.
- Convertus. Convertus kursplaneöversättare. URL <http://www.convertus.se/Kurssida/>. Available 2010-05-19.
- Lambros Cranias, Harris Papageorgiou, and Stelios Piperdis. A Matching Technique in Example-Based Machine Translation. In *Proceedings of the 15th conference on Computational linguistics*, volume 1, pages 100–104, Kyoto, Japan, 1994.
- Eva Ejerhed. Finite state segmentation of discourse into clauses. In A. Kornai, editor, *Extended Finite State Models of Language*, chapter 13. Extended Finite State Models of Language, Cambridge, UK, 1999.
- Eva Ejerhed and Gunnel Källgren. *Stockholm Umeå Corpus. Version 1.0*. Department of Linguistics, Umeå University and Department of Linguistics, Stockholm University, 1997.
- Eva Ejerhed and Daniel Ridings. PAROLE -> SUC and SUC -> PAROLE, 1997. URL <http://spraakbanken.gu.se/parole/tags>. Available 2010-05-19.
- Laurie Gerber and Eduard Hovy. Improving Machine Translation Quality by Manipulating Sentence Length. In David Farwell, Laurie Gerber, and Eduard Hovy, editors, *Machine Translation and the Information Soup: Proceedings of the 3rd Conference of the Association for Machine Translation in the Americas (AMTA'98)*, pages 448–460, Heidelberg, Germany, 1998.
- Jesús Giménez and Lluís Màrquez. Fast and accurate part-of-speech tagging: The SVM approach revisited. In *Conference on Recent Advances in Natural Language Processing (RANLP)*, 2003.
- Péter Halácsy, András Kornai, and Csaba Oravecz. Hunpos - an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, volume Companion Volume, Proceedings of the

- Demo and Poster Sessions, pages 209–212, Prague, Czech Republic, 2007. Association for Computational Linguistics.
- John W. Hutchins and Harold L. Somers. *An introduction to machine translation*. Academic Press, London, England, 1992.
- Andrew Joscelyne. Post-editing: Update on best practices. Translation Automation User Society (TAUS) Report, October 2008.
- Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, New Jersey, 1<sup>st</sup> edition, 2000.
- Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Pearson Education, New Jersey, 2<sup>nd</sup> edition, 2009.
- Suong Dong Kim, Byoung-Tak Zhang, and Taek Kim Yung. Reducing parsing complexity by intra-sentence segmentation based on maximum entropy model. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics*, volume 13, pages 164–171, Hong Kong, 2000.
- Yu-Seop Kim and Yu-Jin Oh. Intra-sentence segmentation based on support vector machines in English-Korean machine translation system. *Expert Systems with Applications*, 34:2673–2682, 2008.
- Vladislav Kuboň, Markéta Lopatková, Martin Plátek, and Patrice Pognan. A Linguistically-Based Segmentation of Complex Sentences. In *Proceedings of FLAIRS Conference*, pages 368–373, 2007.
- Alon Lavie and Abhaya Agarwal. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation at the 45th Meeting of the Association for Computational Linguistics (ACL-2007)*, pages 228–231, Prague, Czech Republic, 2007.
- Vilson Jose Leffa. Clause processing in complex sentences. In *Proceedings of the First International Conference on Language Resources and Evaluation*, volume 2, pages 937–943, Granada, Spain, May 1998.
- Wei-Chuan Li, Tzusheng Pei, Bing-Huang Lee, and Chuei-Feng Chiou. Parsing long english sentences with pattern rules. In *Proceedings of the 13th conference on Computational linguistics*, volume 3, pages 410–412, Helsinki, Finland, 1990.
- Caroline Lyon and Bob Dickerson. Reducing the Complexity of Parsing by a Method of Decomposition. In *International Workshop on Parsing Technology*, pages 215–222, Boston, Massachusetts, 1997.
- Beáta Megyesi. *Data-Driven Syntactic Analysis - Methods and Applications for Swedish*. PhD thesis, Department of Speech, Music and Hearing, Kungliga Tekniska Högskolan, Stockholm, Sweden, 2002.

- Beáta Megyesi. The Open Source Tagger HunPoS for Swedish. In *Proceedings of the 17th Nordic Conference of Computational Linguistics (NODALIDA)*, 2009.
- George Miller and Christiane Fellbaum. Wordnet, 2007. URL <http://wordnet.princeton.edu/>. Available 2010-05-19.
- Constantin Orăsan. A hybrid method for clause splitting in unrestricted english texts. In *Proceedings of the International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications (ACIDCA'2000)*, pages 129–134, Monastir, Tunisia, 2000.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 311–318, Philadelphia, Pennsylvania, July 2002.
- Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In Karel Pala Petr Sojka and Ivan Ivan Kopecek, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Univerisity of Pennsylvania, 1996.
- Advaith Siddharthan. Syntactic simplification and text cohesion. Technical Report 597, University of Cambridge, Cambridge, UK, 2004.
- Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, 2006.
- Ulf Teleman, Staffan Hellberg, and Erik Andersson. *Svenska Akademiens grammatik*, volume 2: Ord. Svenska Akademien, 1999.
- Erik F. Tjong, Kim Sang, and Hervé Déjean. Introduction to the CoNLL-2001 Shared Task: Clause Identification. In *Proceedings of the Fifth Workshop on Computational Language Learning (CoNLL-2001)*, pages 53–57, Toulouse, France, July 2001.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, 2003.
- David Turcato, Fred Popowich, Paul McFetridge, Devlan Nicholson, and Janine Toole. Pre-processing Closed Captions for Machine Translation. In C. Van Ess-Dykema, C. Voss, and F. Reeder, editors, *Proceedings of the Workshop on Embedded Machine Translation Systems*, Seattle, Washington, May 2000.
- Larry Wall and the Perl 5 development team. perlre: Perl regular expressions, 2006. URL <http://perldoc.perl.org/perlre.html>. Available 2010-05-19.

Per Weijnitz. Uppsala Chart Parser Light. Improving Efficiency in a Chart Parser. Master's thesis, Computational Linguistics. Language Engineering Programme. Uppsala University, 2002.

Per Weijnitz, Anna Sgvall Hein, Eva Forsbom, Ebba Gustavii, Eva Pettersson, and Jrg Tiedemann. The machine translation system MATS - past, present & future. In *Proceedings of RASMAT'04 (Recent Advances in Scandinavian Machine Translation)*, Uppsala, Sweden, April 2004.

Fei Xia and Michael McCord. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-2004)*, pages 508–514, Geneva, Switzerland, August 2004.