UPPSALA
UNIVERSITET

Automatic Generation of Post-Editing Rule Sets from Parallel Corpora

Martin Kjellin

Uppsala University
Department of Linguistics and Philology
Språkteknologiprogrammet
(Language Technology Programme)
Bachelor's Thesis in Language Technology, 15 credits

2nd June 2012

Supervisors:
Jörg Tiedemann, Uppsala University
Eva Pettersson, Convertus AB and Uppsala University
Anna Sågvall Hein, Convertus AB and Uppsala University

Abstract

Machine translations often have to be manually post-edited in order to meet quality demands. Since this is a labour-intensive task, it is desirable to find ways to automate (parts of) the post-editing process. This thesis presents a method for generation of rule sets for automatic post-editing of machine translations. The basic idea of the method is to generate a set of candidate rules from a parallel corpus of raw machine translations and manually post-edited translations, and to filter this set by measuring the effect of applying its individual rules to the raw machine translations of another corpus. As indicated by three frequently used evaluation metrics (BLEU, Meteor, and TER), rule sets created using this method are capable of increasing the translation quality of previously unseen translations from the same domain as the corpus used for filtering.

Sammandrag

Maskinöversättningar måste ofta efterbehandlas manuellt för att uppfylla uppställda kvalitetskrav. Eftersom detta är en arbetskrävande uppgift är det önskvärt att hitta sätt att automatisera (delar av) efterbehandlingsprocessen. Denna uppsats presenterar en metod för generering av regeluppsättningar för automatisk efterbehandling av maskinöversättningar. Metoden bygger på att man skapar en uppsättning kandidatregler utifrån en parallellkorpus av råöversättningar och manuellt efterbehandlade översättningar och filtrerar denna uppsättning genom att mäta effekten av att tillämpa dess enskilda regler på råöversättningar från en annan korpus. Tre ofta använda utvärderingsmått (BLEU, Meteor och TER) visar att regeluppsättningar som skapats med denna metod kan öka översättningskvaliteten hos tidigare osedda översättningar från samma domän som den korpus som använts för filtrering.

Contents

Preface

This thesis has only one author, but it would never have come about without help from several others. My supervisors Jörg Tiedemann, Eva Pettersson, and Anna Sågvall Hein have been both helpful and patient in providing me with advice about the study and the thesis. Furthermore, Eva and Anna were the ones who originally suggested the topic of the study. Per Weijnitz has willingly helped me with different technical problems. Sharing workroom with Anton Eriksson and Lina Stadell has given me the opportunity to get involved in many interesting discussions – as well as in a few small controversies on whether to leave the window open or not. Last but not least, Elin Logara has proof-read the thesis and given me valuable suggestions on how to improve its language. I thank them all for their help and support.

1 Introduction

Over the years, joint efforts by large numbers of researchers in language technology have significantly increased the quality of translations produced by machine-translation systems. Still, however, machine translations often require *manual post-editing* in order to meet quality demands. This means that humans adjust the translations and also, implicitly or explicitly, approve those parts of the translations for which no adjustments are required. This method might result in high-quality translations, while requiring less human labour than the traditional, fully manual translation process. Even so, manual post-editing is labour-intensive and might also be tedious and repetitive, given that machine-translation systems tend to repeat their mistakes. Therefore, it is desirable to find ways to automate (parts of) the post-editing process, and consequently, several researchers have studied issues concerning the development of systems for *automatic post-editing* in recent years (see section 2.2). These studies generally show that automatic post-editing can indeed increase translation quality and thereby reduce the need for human labour.

1.1 Purpose

The work that I present in this thesis has been performed at Convertus AB, a language technology company that is a spin-off from Uppsala University (Uppsala, Sweden). One of the company's products is the Syllabus Translator, a system specialized in translating university syllabi from Swedish to English. The system, that is in use at several Swedish universities, contains a module for automatic post-editing. *Raw machine translations* (translations that have not been post-edited in any way) pass this module and are thus transformed into *automatically post-edited translations*. The rules that are used in the module have been developed manually. Currently, Convertus AB cooperates with a number of other language technology companies in the project Bologna Translation Service.[1] The aim of the project is to develop an enhanced syllabus translation system capable of handling more language pairs.

This study is part of the process of developing a post-editing module for the enhanced system. The purpose of the study is to investigate if it is possible to use *parallel corpora* consisting of raw machine translations and *manually post-edited translations* of the same *source sentences* to automatically generate sets of post-editing rules that are capable of increasing the quality of other machine translations. The manually post-edited translations that will be used have been produced by users of the current system. Put in a more concrete

[1] www.bologna-translation.eu

way, my work includes both the generation of two sets of post-editing rules and the development of a general set-generation method. An already existing program (Weijnitz, 2010) will be used for the generation of individual rules, while my contribution to the set-generation method will concern the selection and ordering of rules generated by this program. Generated rule sets will be applied to *test corpora* of previously unseen raw machine translations using a rule-application program that was developed along with the rule-generation program. Translation quality will be assessed through automatic evaluation using three well-established evaluation metrics.

One might ask why automatic post-editing should be employed in efforts to increase the quality of machine translations. Obviously, insufficient translation quality indicates a need for improvements of a machine-translation system. Therefore, it might seem like a detour not to improve the system itself, but to keep the suboptimal system, trying to improve its translations through automatic post-editing. However, there are three main circumstances that motivate this study. First, improving the basic translation system might be a hard task. For example, there is always a risk that new translation errors arise as a result of the correction of old ones. Second, as mentioned above, it has been shown that translation quality can be improved by automatic post-editing. The successful manual development of rules for the post-editing module of the Syllabus Translator provides more evidence for the usefulness of automatic post-editing. As improvements of translation quality are always welcome, these findings motivate further investigations of automatic post-editing, even if the ultimate goal of machine-translation research might be to construct a translation system for which no post-editing is required. Third, users of the Syllabus Translator have produced a large number of manually post-edited translations. This means that much information about translation problems and manual post-editing is readily available. Finding ways to make use of such information in automatic post-editing is an interesting research goal in itself.

I will present a method for rule-set generation whose basic idea is to create a set of candidate rules from a parallel corpus (the *generation corpus*) and then filter (clean) this set by measuring the effect of applying its individual rules to the raw machine translations of another corpus (the *filter corpus*). Those rules that have a beneficial effect on the latter corpus are kept, while the other rules are discarded. I will show that rule sets created using this method are indeed capable of increasing the translation quality of a test corpus, provided that the *domain* (subject field) of the filter corpus (and, optionally, the generation corpus) is the same as the domain of the test corpus.

1.2 Outline

This thesis is organized in the following way. Chapter 2 gives a background to the rest of the study. The evaluation metrics that will be used are discussed, as well as some previous studies in the field of automatic post-editing. The chapter also includes a more detailed account of the Syllabus Translator and the programs for generation and application of post-editing rules. Chapter 3 describes the method for rule-set generation and an implementation of it. Furthermore, corpus data are analysed in order to provide a picture of the errors

that can be found in translations produced by the current system. In chapter 4, two different rule sets that have been generated using the set-generation method are evaluated. Finally, in chapter 5, the results of the study are discussed and some suggestions about enhancements of the method and the implementation are given.

2 Background

The aim of post-editing of machine translations is to improve translation quality. Therefore, this background chapter will start with a discussion of how to evaluate machine translations and, thereby, the systems that have produced them. Manual evaluation will be described briefly, and three important metrics for automatic evaluation, as well as the question of statistical significance of differences in such metrics, will be discussed in detail. Then, some previous studies in the field of automatic post-editing will be summarized. The chapter will close with an account of the Syllabus Translator and two programs for generation and application of post-editing rules that have been developed at Convertus AB.

2.1 Evaluation of Machine Translations

Translations are generally produced to be read by humans. Therefore, letting humans assess translation quality might seem like the ideal way to evaluate machine translations and machine-translation systems. Several approaches to manual evaluation exist. For example, evaluators might be asked to rate the *fluency* and *adequacy* of translations. A fluency rating indicates how intelligible and natural a translation is, while an adequacy rating shows how much of the information in the source sentence that is preserved in the translation (Jurafsky and Martin, 2009, p. 930–931). The underlying assumption is that a translation might be disfluent and still preserve the information in the source sentence, but in practice, evaluators seem to have a hard time separating fluency and adequacy (Callison-Burch et al., 2007). In the context of post-editing, Lagarda et al. (2009) proposed evaluating the *suitability* of machine translations. The idea is that a machine translation should be considered suitable if translators believe that manual post-editing of the translation would require less effort than manual translation from scratch. If not, the translation should be considered not suitable.

Regardless of the approach, however, manual evaluation is both time-consuming and expensive. Therefore, researchers have developed a number of metrics for automatic evaluation of machine-translation quality. The goal of research in this field is to find evaluation metrics that correlate with human judgements of translation quality. Therefore, metrics are usually based on the generally accepted intuition that a *candidate translation* (produced by a machine-translation system) is of higher quality than another if it is more similar to one or several manually produced *reference translations*. Ideally, several reference translations should be used. This is because a certain source sentence

can often be correctly translated in many ways (Jurafsky and Martin, 2009, p. 931). In this study, however, only one reference translation will be used. This is rather common in practice, as it might be hard or impossible to obtain more than one reference translation. The difference between evaluation metrics is in how they compare candidate translations to reference translations.

In order to get several opinions on translation quality, three different metrics will be used in this study, namely *BLEU* (Papineni et al., 2002), *Meteor* (Banerjee and Lavie, 2005; Denkowski and Lavie, 2011), and *TER* (Snover et al., 2006). BLEU is a standard evaluation metric in machine translation and builds upon the well-known concept of *precision*. In the context of machine translation, precision is defined to be the number of n-grams (sequences of n words) in a candidate translation that also occur in any of the reference translations, divided by the total number of n-grams in the candidate translation. Meteor is a somewhat more complex metric that has been developed in response to a number of perceived weaknesses pertaining to BLEU. TER, finally, is designed to measure the amount of editing needed to transform a candidate translation into a corresponding reference translation. Therefore, this metric is especially suited for evaluating progress in replacing manual post-editing with automatic.

2.1.1 BLEU

BLEU (BiLingual Evaluation Understudy), developed by Papineni et al. (2002), is probably the most widely used metric for automatic evaluation of machine-translation quality. The metric uses a variation of precision called the modified n-gram precision, that is calculated as follows. First, find the maximum reference count, that is, the maximum number of times each unique n-gram of the candidate translation occurs in any single reference translation. Then, count the number of occurrences in the candidate translation for each n-gram that also occurs in any of the reference translations. Clip the latter counts, that is, let no count be higher than the corresponding maximum reference count. Finally, sum the counts for all n-grams and divide the sum by the total number of n-grams in the candidate translation. To find the modified n-gram precision for a set of sentences, add the clipped n-gram counts for all sentences in the candidate translation and divide the resulting figure by the total number of n-grams in the candidate translation. The result is the modified precision p_n.

The next step in the calculation of a BLEU value is to combine the modified precisions for different n. Take the logarithm of each p_n, multiply by a weight w_n (usually set to $1/N$ for all n, where N is the maximum n-gram length), sum over all n, and finally take the exponential of the result. This gives the *geometric mean* G of the modified precisions:

$$G = \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Modified precision is generally high for candidate translations that are shorter (in number of words) than the reference translations. However, such candidate translations might fail to preserve all information. Therefore, BLEU also has a *brevity penalty* B. Call the length of the reference sentence closest in length to a candidate sentence the best match length. Sum all the best match

lengths for all sentences in the candidate translation to obtain the effective reference length r. Denote the total length of the candidate translation by c. Then, calculate the brevity penalty using the formula

$$B = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leqslant r \end{cases}$$

Finally, calculate the BLEU score by multiplying B by G. This yields BLEU scores in the range from 0 to 1, where higher scores are supposed to indicate higher translation quality.

As mentioned, BLEU is widely used for machine-translation evaluation. However, it has also been severely criticized. In particular, researchers have questioned whether BLEU correlates as well with human judgements of translation quality as it was initially believed to do. Callison-Burch et al. (2006) show that, under certain conditions, candidate translations could be heavily altered (through reordering and substitution of words) with no effect on their BLEU scores. Therefore, BLEU might be unable to distinguish between translations of different quality. Callison-Burch et al. note that a consequence of this is that an increase in BLEU score might not indicate an actual improvement of translation quality. Furthermore, they find that BLEU might give too low scores to rule-based machine-translation systems (as compared to statistical ones). This finding is confirmed by Koehn and Monz (2006). Callison-Burch et al. recommend the use of BLEU only for certain tasks. For example, BLEU might be used for comparisons of systems of the same type or for evaluation of changes to a certain system.

Banerjee and Lavie (2005) list four important weaknesses of BLEU. First, they note that BLEU is based on precision and only indirectly, through the brevity penalty, considers *recall* (the number of n-grams found in both a candidate translation and a corresponding reference translation, in proportion to the total number of n-grams in the reference translation). They show figures indicating that recall in fact correlates better than precision with human judgements of translation quality. Thus, inclusion of recall in evaluation metrics would improve their quality. Second, like Callison-Burch et al. (2006), the authors note problems concerning word order or grammaticality. Like recall, grammatical wellformedness is measured only indirectly by BLEU, through the use of higher-order n-grams. Third, the n-gram counts used in BLEU do not require an explicit word-to-word matching between candidate translation and reference translation. That is, a certain word might occur in both translations but correspond to different words (different occurrences of a word) in the source sentence. Fourth, if the modified precision for some n is zero, so is the BLEU score, even though the precision for shorter n-grams might be positive. Therefore, Banerjee and Lavie hold that BLEU scores at the sentence level might be meaningless. However, they acknowledge that BLEU was never intended for evaluation of translations of individual sentences.

Two different implementations of BLEU will be used in this study. The first of these is *Multi-BLEU* (revision 3612), a script that is included in the machine-translation system Moses (Koehn et al., 2007), while the second, that will be referred to as *ZV-BLEU*, is part of a set of scripts developed by Zhang and Vogel (2004) that calculate confidence intervals and test statistical significance

for evaluation scores (see section 2.1.4). For both, the maximum n-gram length N will be 4. This is a common practice that was also used by Papineni et al. (2002) in the original presentation of BLEU.

2.1.2 Meteor

According to Banerjee and Lavie (2005), Meteor (Metric for Evaluation of Translation with Explicit ORdering) was designed to avoid the four weaknesses of BLEU that were described in section 2.1.1. The lack of word-to-word matching found in BLEU is avoided by the creation of an *alignment* between candidate translation and each reference translation. The alignment is constructed from possible matches (with respect to certain criteria) between words and phrases in the two translations. The most recent version of the metric and the one that will be used in this study, Meteor 1.3 (Denkowski and Lavie, 2011), uses three methods for matching words and one for matching phrases. Words can be matched if they have the same surface form, if they have the same stem, or if they are considered to be synonyms. Phrases can be matched if they are considered to be paraphrases of each other. The alignment between the translations is the largest possible subset of the identified matches that meet four ranked criteria. First, each candidate-translation word should match one or zero reference-translation words, and vice versa. Second, the number of matched words in both translations should be maximized. Third, the number of *chunks* should be minimized. In this context, a chunk is a (maximum-size) group of matches that are adjacent in both translations. Fourth, the sum of the distances between matched words and phrases in the translations should be minimized. That is, matched words and phrases should preferably occur at about the same place in candidate translation and reference translation.

Denkowski and Lavie describe how a Meteor score is calculated from an alignment. The calculation involves a number of parameters, α, β, γ, δ, and $w_1 \dots w_n$. These parameters can be tuned in order to make the metric correlate with human judgements. For each matching method m_i, the number of matched content and function words in the candidate translation, $m_i(h_c)$ and $m_i(h_f)$, as well as the number of matched content and function words in the reference translation, $m_i(r_c)$ and $m_i(r_f)$, is counted. The same thing is done for the total number of content and function words in the candidate translation, $|h_c|$ and $|h_f|$, and the reference translation, $|r_c|$ and $|r_f|$. Weights w_i for the different matching methods and a content–function word weight δ are then used to calculate *weighted precision* P and *weighted recall* R:

$$P = \frac{\sum_i w_i \cdot (\delta \cdot m_i(h_c) + (1-\delta) \cdot m_i(h_f))}{\delta \cdot |h_c| + (1-\delta) \cdot |h_f|}$$

$$R = \frac{\sum_i w_i \cdot (\delta \cdot m_i(r_c) + (1-\delta) \cdot m_i(r_f))}{\delta \cdot |r_c| + (1-\delta) \cdot |r_f|}$$

From these, a *parameterized harmonic mean* F is calculated:

$$F = \frac{P \cdot R}{\alpha \cdot P + (1-\alpha) \cdot R}$$

A *fragmentation penalty* Q incorporates gaps and differences in word order into the metric. This penalty is based on the number of matched words m (averaged

over candidate translation and reference translation) and the number of chunks d. The penalty grows with the number of chunks:

$$Q = \gamma \cdot \left(\frac{d}{m}\right)^{\beta}$$

Finally, the Meteor score M can be calculated:

$$M = (1 - Q) \cdot F$$

As in the case of BLEU, higher scores are supposed to indicate higher translation quality. Banerjee and Lavie (2005) add that in case several reference translations are used, the candidate translation is scored against each of these, and the best score is reported.

2.1.3 TER

The TER (Translation Edit Rate or Translation Error Rate) metric, described by Snover et al. (2006), uses a different approach to machine-translation evaluation than BLEU and Meteor. TER intends to measure the necessary amount of manual post-editing of machine translations. It is defined as the minimum number of certain editing operations (described below) that is needed to make a candidate translation match one of the available reference translations, normalized (divided) by the average length (in number of words) of the reference translations. Thus, it measures the *edit distance* between the candidate translation and the closest reference translation. The lower the TER is, the closer the candidate translation is to the reference translation. Snover et al. (2006) believe that TER is an easily understandable evaluation metric, even for non-professionals. Furthermore, they state that TER requires fewer reference translations than BLEU to achieve the same correlation with human judgements of translation quality.

TER takes four different editing operations into account. Three of these, insertion, deletion, and substitution, concern single words, while the fourth, shift, operates on word sequences. Shifting means moving a contiguous sequence of words from one place to another within the candidate translation. The number of editing operations is calculated in two steps. First, the shifts are found by repeated selection of the shift that results in the greatest reduction of insertions, deletions, and substitutions (until no beneficial shifts remain). Second, the minimum number of these three operations needed to complete the transformation is calculated using dynamic programming, a problem-solving method in which solutions to subproblems are saved in a table and can be used several times in the process of solving the original problem (Cormen et al., 2009, p. 359). TER is computed using a program called TERcom. In this study, TERcom version 0.7.25 will be used.

2.1.4 Statistical Significance

The use of different metrics in automatic evaluation of machine-translation quality raises the question of *statistical significance*. That is, given a difference in scores between two translations (for example, a raw machine translation

and an automatically post-edited translation), how likely is it that there is a true difference in performance (as indicated by the evaluation metric) between the systems that produced the translations? A result, such as a difference in scores, is said to be statistically significant if the probability that the result has occurred by chance is lower than a predetermined threshold, called the significance level (Sandberg and Sandberg, 2002, p. 41). The reason to consider statistical significance is that machine-translation systems must necessarily be evaluated on samples of sentences (test corpora), while the purpose of such an evaluation is to assess the performance of a system on all possible sentences from a certain domain (Koehn, 2004), or even in general. Thus, the 'true' score for the system is unknown.

Koehn (2004) and Zhang and Vogel (2004) advocate the use of *bootstrapping* (also called *bootstrap resampling*) to find *confidence intervals* for evaluation scores. A confidence interval is a range of scores that is believed to include the true score. The idea of the bootstrapping technique is to construct artificial test corpora by sampling sentences from a parallel corpus. The sampling is done with replacement, which means that a sentence from the original corpus may occur zero, one, or several times in a certain constructed corpus. Thus, it is possible to construct a large number of test corpora of the same size as the original corpus. These corpora will be different from both each other and the original corpus. The scores for the corpora are then calculated and sorted, and the lowest and highest scores are dropped. The number of scores to drop depends on the total number of test corpora and the desired confidence interval. A common practice is to estimate a 95 % confidence interval, in which case the middle 95 % of the scores should be kept, or, equivalently, the top 2.5 % and the bottom 2.5 % should be dropped. The result is a confidence interval delimited by the lowest and highest of the kept scores. Koehn provides some empirical evidence that this method for estimating confidence intervals is as good as estimation from the scores for a large number of independent test corpora.

Zhang and Vogel (2004) describe how bootstrapping can be used to assess the statistical significance of score differences between two machine-translation systems. For each of the constructed corpora, the score difference between the two systems is calculated. Then, the obtained figures are sorted, and the middle 95 % are kept. If the resulting confidence interval does not include 0 (that is, if all score differences in the confidence interval are either less than or greater than 0), the conclusion can be drawn that the difference between the systems is statistically significant (given a desired significance level of 5 %). Zhang and Vogel have implemented this technique in a set of scripts that will be used in this study. The same authors studied a number of questions related to statistical significance. They found that differences in BLEU scores between different systems remained stable when the test corpus size was increased beyond about 250 sentences. However, larger corpora resulted in narrower confidence intervals (more discriminative scores). Likewise, more reference translations meant narrower confidence intervals. An important finding was that larger test corpora can compensate for fewer reference translations, as regards the width of confidence intervals. Finally, Zhang and Vogel investigated how many bootstrap samples (constructed test corpora) that are needed in order to get reliable confidence intervals, and found that there are only small changes in the intervals when the number of samples is increased beyond 2,000.

2.2 Machine Translation and Post-Editing

Researchers in language technology have developed a large number of different methods for machine translation. A coarse-grained but useful characterization of these methods can be obtained through the distinction between *rule-based* and *statistical* approaches to machine translation. The linguistic rules that have given rule-based machine translation its name are generally constructed manually. Rule-based machine-translation systems differ in how deep a linguistic analysis they perform (Jurafsky and Martin, 2009, p. 903–910). *Direct translation* involves little more than word-by-word translation by means of a bilingual dictionary. In *transfer* approaches, the translation process is divided into three phases. In the *analysis* phase, parsing of the source sentences results in representations of their syntactic structure. Then, in the *transfer* phase, the syntactic representations are transformed into corresponding target-language representations (*syntactic transfer*), and the words are translated by means of a bilingual dictionary (*lexical transfer*). Finally, in the *generation* phase, translations (raw machine translations) are produced from the target-language representations. In *interlingua* approaches to machine translation, language-independent meaning representations are constructed from the source sentences. From these representations, the translations are produced.

As for statistical machine-translation systems, they do not (in their pure form) require any manually encoded linguistic knowledge. Instead, these systems build upon the idea of finding a most probable translation given a *language model* of the target language and a *translation model* of the relationship between the source language and the target language. This approach presupposes the existence of parallel corpora where the same texts are present in both the source language and the target language. The translation model is constructed by *alignment* of source-language and target-language words and phrases (Jurafsky and Martin, 2009, p. 910–930).

Several authors, for example Knight and Chander (1994), Simard et al. (2007a), Isabelle et al. (2007), and Lagarda et al. (2009), have noted that post-editing of output from machine-translation systems can be considered a translation process in itself. Viewed in this way, post-editing means that a text written in a 'raw machine-translation language' is translated into a 'post-edited language'. Therefore, it comes as no surprise that the main approaches to machine translation have also been applied to the automatic post-editing task. In this section, some previous studies within the rule-based and the statistical approach to automatic post-editing will be summarized.

2.2.1 Rule-Based Post-Editing

Linguistic analyses in rule-based post-editing are often shallow or non-existing. This means that this approach to post-editing can be seen as a kind of direct translation. An early study in ruled-based post-editing of machine translations was performed by Knight and Chander (1994). In the context of translation from Japanese to English, they developed rules for inserting missing articles. Training data consisted of English texts from which articles had been removed, so that it was known from the start where articles should be inserted. Decision trees for choosing between *the* and *a/an* were automatically built. These trees

were based on features of the context. The features were both lexical (for example, what word preceded the article) and grammatical (for example, tense or part of speech). Knight and Chander report accuracy figures (78 %) that approach those reached by humans (83–88 %).

Stymne and Ahrenberg (2010) note that because statistical machine-translation systems include little or no linguistic knowledge, their output often contains grammatical errors. Therefore, they investigated if a mainly rule-based grammar checker, designed to find errors in texts written by humans, could be used for automatic post-editing of output from such a system, trained to translate from English to Swedish. The grammar checker can identify and categorize suspected errors. In order to make the most out of the checker as a post-editing tool, Stymne and Ahrenberg first studied for which error categories the checker produces good correction suggestions. When testing the system, correction suggestions were accepted if they belonged to categories with a large majority of good correction suggestions. According to evaluations using BLEU and TER, translation quality was increased by the accepted changes, but the improvements were small. This was related to the fact that only a small share of the sentences (and the actual errors) were changed in the post-editing process. Some of the changes were also evaluated manually, and it was found that more than 70 percent of them were improvements. Stymne and Ahrenberg hope that the results can be improved by extensions of the rule set, by integration between the machine-translation system and the grammar checker, or by the development of a checker specialized in handling machine-translation errors.

Allen and Hogan (2000) used a parallel corpus of raw machine translations and manually post-edited translations to automatically extract post-editing rules correcting errors in raw machine translations from English to French, and vice versa. Each rule prescribed the adjustment of a string (one or several words) in a certain context of strings. The latter strings might be empty, so that the adjustment is always carried out, regardless of the context. Allen and Hogan note that the most frequent incorrect constructions should be identified, but they do not describe how rules are selected, nor do they evaluate the generated rules. However, they do discuss the risk of hypercorrection, that is, that an acceptable raw machine translation might be incorrectly altered into a form that must be manually post-edited back to the original one. They hold that some hypercorrections are acceptable if automatic post-editing can handle most of the necessary adjustments and thus reduce the need for human labour.

Guzmán (2007) advocates automation of post-editing by means of formulae, so-called *regular expressions*, that characterize sets of strings (Jurafsky and Martin, 2009, p. 51–59). The regular expressions are designed to locate different types of errors (spelling, punctuation, agreement, word order, and so on) that might occur in machine translations from English to Spanish. Guzmán holds that the regular expressions should be flexible (capable of correcting as many potential errors as possible), and that both frequency and required time for manual post-editing are important factors in choosing what errors to correct automatically.

Elming (2006) used *transformation-based learning* (TBL) to construct a set of post-editing rules for correcting output from a rule-based machine-translation system. TBL is a machine learning algorithm that has been applied to several different problems in natural language processing. Brill (1995) describes how

the TBL algorithm constructs a list of so-called *transformations* (rules), that can easily be interpreted by humans. First, a so-called initial-state annotator is applied to a corpus of unannotated text. This annotator is some more or less sophisticated system for finding part-of-speech tags, syntactic structures, corrections, or some other linguistic information relevant to the task at hand. The output of the initial-state annotator is then compared to another annotation that is considered to be true. Using manually predetermined transformation templates, the algorithm repeatedly finds the single transformation that would bring the current annotation as close as possible to the true one. This transformation is added to an ordered list, and the corpus is updated according to the transformation.

As input to the learning algorithm, Elming used a parallel corpus of machine translations from English to Danish and manually post-edited versions of these translations. The machine translations were part-of-speech tagged and word-aligned to their manually post-edited counterparts. The only correction type investigated by Elming was substitution. In this case, substitution included addition of words (substitution of nothing) and deletion of words (substitution with nothing). Elming constructed 70 transformation templates to be automatically instantiated in the study. In general, these rule templates prescribed that word A should be replaced by word B if word C or part-of-speech tag D was found in certain sets of preceding and following positions. For a rule to be included in the list, it had to pass an accuracy threshold (be successful in at least half of its corrections) and a score threshold (make at least three more good corrections than bad corrections). The original machine-translation output and the automatically post-edited version of a test data set were compared to a manually post-edited reference translation using BLEU. Elming found that the automatic post-editing increased the translation quality (according to BLEU), but also that a majority of the constructed rules were found not to be general enough to apply to unseen data.

2.2.2 Statistical Post-Editing

Several authors have explored the approach of using a statistical machine-translation system to post-edit the output of a rule-based system. The systems are trained on parallel corpora of raw machine translations, on the one hand, and manually post-edited translations or manual from-source translations, on the other. Such a combination of rule-based and statistical machine translation can be seen as a way to 'take advantage of the particular capabilities of each system' (Lagarda et al., 2009, p. 217).

The idea of an automatic post-editing module based on statistical machine-translation techniques was mentioned already by Knight and Chander (1994), but Simard et al. (2007a) claim to be the first to actually implement it. According to BLEU and TER scores obtained by Simard et al., a combination of rule-based translation (from French to English and vice versa) and statistical post-editing outperformed both a plain rule-based system and a statistical system trained to translate directly from the source language. While the authors consider their results very encouraging, they note that, when given more data, the output quality of the statistical from-source translation system increased faster than that of the combined system. This suggests that with sufficient

amounts of data available, the former system would outperform the latter.

In a follow-up study, Simard et al. (2007b) examined whether the promising results of Simard et al. (2007a) were due to the fact that the raw machine translations and the manually post-edited translations were not produced independently of one another (the latter being edited versions of the former). According to Simard et al., this type of data is seldom available, especially not in larger quantities, and therefore, they investigated the impact of using target-language data from standard parallel corpora in the training process instead of manually post-edited machine translations. The corpora contained proceedings of the European Parliament and news commentary texts. The texts were translated from English to French, and vice versa. As in the earlier study, a system using statistical post-editing of rule-based translations outperformed both a plain rule-based system and a plain statistical one when using smaller quantities of training data. This indicates that 'source' and 'target' training data for the post-editing component need not necessarily be directly connected. Another interesting finding of Simard et al. (2007b) is that pure statistical translation using large amounts of data does not, as Simard et al. (2007a) believed, outperform the post-editing approach in terms of BLEU scores. Indeed, statistical translation catches up once a sufficient amount of data is used, but if the amount is increased beyond this point, the two approaches seem to stay equivalent.

Dugast et al. (2007) performed a qualitative analysis of the results reported by Simard et al. (2007b) and of a similar experiment in post-editing of translations from German and Spanish to English. Changes made in the post-editing process were divided into categories (different lexical and grammatical changes, punctuation, word order, etcetera). For each category, improvements and degradations caused by the post-editing were counted, and the figures were compared to an acceptance criterion requiring at least eight improvements for every degradation. Despite the positive overall impact of the post-editing, the acceptance criterion was not met.

The same group of researchers that performed the studies reported by Simard et al. (2007a) and Simard et al. (2007b) also studied automatic post-editing in the context of domain adaptation of machine-translation systems (Isabelle et al., 2007). The domain in question was job advertisements (translated from French to English and vice versa). Translations from a generic rule-based machine-translation system were automatically post-edited using a statistical machine-translation system trained on a domain-specific parallel corpus of raw machine translations and manually post-edited translations. The automatically post-edited translations were then compared to translations produced by a version of the rule-based system equipped with domain-specific dictionaries. Such lexical adaptation is a common type of domain adaptation of rule-based machine-translation systems (see section 2.3.1).

Isabelle et al. found that automatic post-editing of output from the generic system outperformed both translation using the lexically adapted system and statistical translation from scratch (in terms of BLEU and TER scores). Post-editing of the output from the lexically adapted system resulted in even better scores, but differences were small between this strategy, on the one hand, and the combination of the generic system and post-editing, on the other. The authors conclude that most of the useful information included in the domain-

specific dictionaries can be extracted from the corpus on which the statistical machine-translation system was trained. Thus, they hold that statistical post-editing could be used for domain-adaptation in place of the labour-intensive dictionary-building strategy.

Yet another example of the technique of using a statistical machine translation system to post-edit output from a rule-based system is given by Lagarda et al. (2009), who applied it to two English-to-Spanish corpora of proceedings from parliamentary sessions and medical protocols, respectively. The results of this combined translation system were compared to those of plain rule-based translation and of statistical translation directly from the source texts. Figures from automatic evaluation using BLEU and TER showed promising results for the combined system. In the case of the protocols, it outperformed both the other systems, and in the case of the proceedings, its results were better than those of the plain rule-based system and not much worse than those of the plain statistical system. Also, manual evaluation indicated that the combined system improved the suitability (see section 2.1) of the translations, at least as regards the proceedings.

2.3 The Convertus Systems

2.3.1 The Syllabus Translator

The Syllabus Translator is an application of the Convertus Hybrid Translator, which is a modular hybrid machine-translation system that follows the rule-based transfer approach (see section 2.2), but that also uses several fall-back strategies for cases not covered by the rules of the different modules (Weijnitz et al., 2004; Sågvall Hein, 2012). The fall-back strategies distinguishes the system from MATS (Sågvall Hein et al., 2002, 2003), the system on which it is based. In turn, MATS is a scaled-up version of the MULTRA system (Beskow, 1993; Sågvall Hein, 1994, 1997). In all these systems, unification of feature structures plays an important role in all three phases of the translation process (analysis, transfer, and generation).

If a source sentence cannot be completely parsed by the system in the analysis phase, the fall-back strategy is to select a set of partial syntactic analyses that together cover the input and then translate each partial analysis independently. As regards the transfer phase, those parts of the syntactic source-language representation for which no transfer rules exist are just copied to the target-language representation (Weijnitz et al., 2004). The dictionaries (see below) that are used in the lexical transfer are supplemented by lexical transfer rules that handle context-dependent translations, such as idiomatic expressions (Pettersson, 2005, p. 14). In other words, a more specific rule (the lexical transfer rule) is preferred over a more general one (the dictionary entry). This specificity principle is generally applied in the transfer and generation phases (Sågvall Hein, 1994). In case the generation-phase rules fail, the system generates a word-by-word translation, placing the translated words in their source-language order (Weijnitz et al., 2004). The most probable translation of each word is found by means of a statistical language model (Sågvall Hein, 2012).

When a translation produced by the system has been manually post-edited

by a user, it is stored in a translation memory that is part of the system. Before a source sentence is translated, the translation memory is consulted. If the source sentence has been translated before, the manually post-edited translation is retrieved from the translation memory. If not, the system translates the source sentence from scratch. The more the system is used, the more translations will be stored in the translation memory and the higher the translation quality will get (Sågvall Hein, 2008).

Since the Syllabus Translator has been designed to translate from Swedish to English, all input to the system is supposed to be written in Swedish. In other words, the system analyses all source sentences, regardless of their language, as if they were written in Swedish (Pettersson, 2011). For obvious reasons, problems will arise if this premise does not hold, something which might happen (see appendix A). Another source of problems for machine-translation systems is the lexical ambiguity that is inherent in natural languages. That is, one word might have several senses, and these might in turn correspond to different words in the target language. In the Syllabus Translator, such situations are handled by means of a dictionary hierarchy. Apart from a general dictionary, this hierarchy includes a dictionary for general education terminology and several dictionaries for terminology from different domains. Here, scientific fields constitute separate domains. If there are several possible translations of a certain word, the system chooses the translation that is found in the most specific of the applicable dictionaries. This strategy increases the probability that ambiguous words will be translated correctly (Sågvall Hein, 2008; Pettersson, 2010).

The developers of the Syllabus Translator have found that the system produces a number of translation errors that are hard to correct by adjustments to the modules for analysis, transfer, and generation, but rather easy to handle through automatic post-editing. Therefore, a post-editing module has been included in the system. An automatically post-edited translation that has been produced by this module might be identical to the corresponding raw machine translation, in case no adjustments are prescribed by the post-editing rules. A user of the system will only see the automatically post-edited translations (Pettersson and Sågvall Hein, 2011).

The set of post-editing rules that is currently in use has been manually developed and is based on experience of the performance of the system. Each rule consists of a left-hand side and a right-hand side that are separated by the characters ->. The regular expression on the left-hand side is matched against the translation to be post-edited. Each matching string in the translation is then replaced according to the right-hand side. This rule set, that will be referred to as rule set I, comprises 1,465 rules. A few of the rules in the set can be found in figure 2.1. For example, the fourth of the rules prescribes the substitution of ‘an’ by ‘a’, if the following word is ‘university’ or ‘University’. In general, rule set I has been found to increase translation quality. In chapter 4, the application of the rule set to two different corpora will be evaluated using BLEU, Meteor, and TER.

Figure 2.1: Five rules from the manually developed rule set I.

```
should can->should be able to

loose((?: [^\s]+){0,2}) (problems?)->solve$1 $2

loose((?: [^\s]+){0,2})( [^\s]*)? (equations?)->solve$1$2 $3

 an ([uU]niversity)-> a $1

the same (.*?) that->the same $1 as
```

2.3.2 Generation and Application of Post-Editing Rules

Two Perl programs for string-based post-editing have been developed at Convertus AB. One of the programs generates post-editing rules from parallel corpora and saves them to a text file with one rule per line, while the other automatically post-edits translations by applying such rule files (Weijnitz, 2010). These programs form the basis of the method for generating post-editing rule sets that has been developed in this study (see chapter 3). The rule-generation program creates rules in a standardized format. Thus, these rules are somewhat different from the manually developed rules that are used by the current post-editing module of the Syllabus Translator (see section 2.3.1). The format is exemplified in figure 2.2.

The input to the rule-application program should (apart from a rule file) be a text file with one raw machine translation per line. Like in the current post-editing module of the Syllabus Translator, the regular expression on the left-hand side of each rule is matched against the translation to be post-edited, and matching strings are replaced according to the right-hand side (Weijnitz, 2010). Apart from the regular expressions, the conditional operator of Perl (Wall et al., 1996, p. 91–92) is used on the right-hand side of each rule to ensure that spaces are inserted where needed. The result of a rule application is the deletion, insertion, or substitution of one or several *tokens*. Here, and in the rest of the thesis, tokens are stretches of characters that are separated from each other with whitespace. Thus, a token can be a word, but also a punctuation mark or some combination of words and punctuation marks. For example, the third rule in figure 2.2 prescribes the substitution of 'various' by 'different', if the preceding context is 'with' and the following context is 'types'. The rules of a rule file are applied in order, in such a way that the first rule is applied to a line of the input, the second rule is applied to the output of this application, and so on, until the last rule has been applied. Thus, not only are the particular rules important for the result of the post-editing process, but so is the order of the rules (Weijnitz, 2010).

A parallel corpus to be used by the rule-generation program should be stored as a text file consisting of line triplets. The second line of each triplet should contain a raw machine translation, and the third should contain a corresponding reference translation (Weijnitz, 2010). The first line of each triplet might contain the corresponding source sentence. However, since source

Figure 2.2: A deletion rule, an insertion rule, and a substitution rule created by the rule-generation program. Lines have been split in order to improve readability.

```
((?:^|\W)this)\s*course\s*(syllabus(?:\W|$))->
                      "$1".("$1"?" ":"")."$2"

((?:^|\W)through)\s*(individual(?:\W|$))->
                      "$1".("$1"?" ":"")."an".("$2"?" ":"")."$2"

((?:^|\W)with)\s*various\s*(types(?:\W|$))->
                      "$1".("$1"?" ":"")."different".("$2"?" ":"")."$2"
```

sentences are not used in the rule-generation process, the first line might also contain something else or even be empty. The reference translation in the third line can be thought of as a manually post-edited version of the raw machine translation in the second. However, the reference translation might in fact have been produced in some other way. For example, it might be a fully manual translation of the source sentence.

The rule-generation program compares raw machine translations to reference translations using an implementation of the *diff* algorithm written by Dominus et al. (2004), who describe the algorithm in the following way. The purpose of diff is to find a smallest set of changes (insertions and deletions) that will turn a sequence of items (here, tokens) into another. This requires the calculation of a *longest common subsequence* (LCS) between the two sequences. An LCS is a longest sequence that can be formed from each of the two sequences through deletion of items. To turn one of the original sequences into the other, what needs to be done is to delete the items that are present in the first sequence but not in the LCS and then insert the items that are present in the second sequence but not in the LCS. There are several LCS algorithms. The one used in this diff implementation is an improved version of an algorithm that was described by Hunt and Szymanski (1977). Note the similarities between the diff problem and the calculation of TER (see section 2.1.3). From the diff output, the rule-generation program produces one post-editing rule for each pair of *chunks* that differs between the raw machine translations and their reference counterparts. Here, and in the rest of the thesis, a chunk is an uninterrupted, maximum-length stretch of tokens, including intervening whitespace characters. Note that this definition is somewhat different from the one used in the description of the Meteor evaluation metric (see section 2.1.2). The default context sizes for the rules are one token before and after each pair of differing chunks, but users may specify larger contexts.

3 Method and Data

The previous chapter included an account of some earlier studies in the field of post-editing of machine translations. It also presented the Syllabus Translator and a program capable of generating post-editing rules from parallel corpora. This chapter will describe the method for generating *sets* of such rules that has been developed in this study. Furthermore, the data that have been used in the study will be presented.

The chapter is divided into five sections. First, an overview of the set-generation method will be given. Second, data consisting of two groups of parallel corpora will be presented. These corpora have been used in the development of the method and in the generation of two sets of post-editing rules that will be described and evaluated in the next chapter. Third, one of the corpora will be analysed in order to provide a picture of the errors that arise in translations produced by the current system. Fourth, a number of details of the set-generation method will be elaborated. Fifth, an implementation of the method will be described.

3.1 Method Overview

A set of post-editing rules comprising one rule for each pair of differing chunks in a parallel corpus could easily be generated using the rule-generation program presented in section 2.3.2. However, such a set-generation strategy is unlikely to result in a rule set capable of improving translation quality. Some of the rules might not be applicable to other translations than those from which they were generated (because the errors on which the rules were based are very uncommon), while other rules might actually decrease translation quality when applied to other translations. The latter problem is the hypercorrection problem discussed by Allen and Hogan (2000) and described in section 2.2.1. For example, a certain pair of differing chunks might give rise to a rule prescribing that 'on' should be replaced by 'in' when preceded by 'written' and followed by 'a'. This replacement might be favourable in some cases (probably, for example, if the raw machine translation were 'written on a book'), but, very likely, not in all ('written on a wall'). Furthermore, the ordering of the rules in the set might be inadequate.

In order to empirically test the assumption that this naïve set-generation strategy does not result in high-quality rule sets, a simple experiment was carried out. The rule-generation program was utilized to generate all possible rules from the pairs of differing chunks between the raw machine translations and the manually post-edited translations of a parallel corpus ('the medical

Figure 3.1: A raw machine translation and the result of post-editing it using an unfiltered rule set. Lines have been split in order to improve readability.

```
account at a general level for various types of cognitive support
for the brukargruppen

- - - - - - at a general level , - give an overview  at a general
level , - give an overview account , at a  at a general level , ,
explain , for  know the skills of different types by means of
previously acquired knowledge base in binocular vision analysis
that is appropriate for the degree project independently
independently  methods included in the course ,  of cognitive
support ) , and the Swedish Optometric Association ( Optikerföbundet )
. based  based  based on ones own , . audiologist  are given .
audiologist  3-4  for basics of acute impact  based  based  based
on  know the brukargruppen
```

generation corpus', to be described in section 3.2). The context sizes of the rules were one token before and one token after each pair of differing chunks. No measures were taken to order the rules or to remove duplicated, deteriorating or useless rules from the generated rule set. The set, comprising 48,454 rules, was applied to a corpus of raw machine translations ('the medical filter corpus', see section 3.2). The result was an obvious degradation of translation quality. The output was certainly not correct sentences, but rather more or less incomprehensible stretches of words. Furthermore, the size of the output (in terms of both characters and tokens) was more than 40 times the size of the input, apparently because of strange repetitions and insertions of words. The low translation quality can also be seen in the (Multi-)BLEU score, that was 0.5634 before the rules were applied, but only 0.0073 afterwards. An example from the experiment is given in figure 3.1.

The above findings provide some empirical support for the assumption that naïvely generated post-editing rule sets are not very useful. Therefore, the work presented here has been concentrated on the development of a method for selection and ordering of post-editing rules produced by the existing rule-generation program. The basic idea of the method is to create a set of candidate rules from a parallel corpus (the generation corpus) and then filter (clean) this set by applying its rules, one by one, to the raw machine translations of another corpus (the filter corpus). Rules that are capable of increasing the BLEU score of the latter corpus (when using manually post-edited translations as reference translations) are kept, while all other rules are discarded. The intention of the filtering is to find a set of rules that are likely to be useful outside the corpus from which they have been generated. The chosen criterion for keeping rules is easy to apply and makes it possible to distinguish between rules of higher and lower quality, but might be unnecessarily strict or weak. On the one hand, if the filter corpus is large enough, a rule that is actually useful might be discarded because it has no observable effect on the BLEU score. On the other hand, an increased BLEU score might be due to peculiarities of the filter corpus and not to the general usefulness of the evaluated rule. The choice of evaluation metric

for the filtering process might also be questioned. As described in section 2.1.1, the BLEU metric has been heavily criticized. Still, it is a standard metric in machine translation that gives a rough estimate of translation quality. This justifies its use here.

The method resembles the transformation-based learning approach of Elming (2006) that was described in section 2.2.1. In both methods, candidate rules are kept if they seem to increase translation quality. However, the method presented here does not include the time-consuming strategy of ordering the rules by repeatedly finding the most error-reducing one. Instead, the ordering of the rules is postponed to a later stage of the process, when all rules to be included in the final rule set are already selected. A possible variation of the described method would be to use the same corpus as both generation corpus and filter corpus. This would probably have the same effect of eliminating many rules that are not generally useful, but might mean that a few rules that are peculiar to the corpus would be kept.

An important aspect of the method is that all candidate rules (and thus, all rules in filtered rule sets) should be based on differences between raw machine translations and manually post-edited translations. An alternative would be to base the rules on differences between translations post-edited using rule set I (see section 2.3.1) and manually post-edited translations. However, because rule set I is supposed to be used for post-editing of raw machine translations, such a strategy would not enable direct comparisons of the post-editing effects of rule set I and new, automatically generated rule sets. Note, however, that even though both rule set I and the new rule sets are specially adapted for post-editing of raw machine translations, nothing precludes the possibility of applying them in series (even though the result might not be optimal).

3.2 Data

The method for rule-set generation that was described in section 3.1 presupposes the existence of at least one generation corpus and at least one filter corpus. Furthermore, at least one test corpus is required for the evaluation of post-editing rule sets generated from these corpora. This section describes two groups of corpora that have been built and used in this study. The contents of the corpora originally come from two translation memories of the Syllabus Translator. The building of the corpora included several steps of cleaning and normalization of the data from the translation memories. These measures are summarized in appendix A.

The first group of corpora consists of a generation corpus, a filter corpus, and a test corpus. These corpora, that contain data from medical syllabi and thus will be referred to as the *medical* corpora, comprise totally 23,174 *segments* (see below for a definition). The generation corpus contains 18,540 segments (80 % of the total number), while the filter corpus and the test corpus contain 2,317 segments (10 %) each. In contrast to this first group of corpora, the second group contains no generation corpus, but consists only of a filter corpus (3,756 segments) and a test corpus (3,755 segments). These two corpora contain data from a broader spectrum of scientific fields than the medical corpora. Thus, they will be referred to as the *general* corpora. Each segment in the corpora

Table 3.1: Tags and corresponding parts of speech.

tag	part of speech
CC	coordinating conjunction
CD	cardinal number
DT	determiner
IN	preposition or subordinating conjunction
JJ	adjective
MD	modal
NN	noun
PR$	possessive pronoun
RB	adverb
RP	particle
TO	*to*
VB	verb
,	comma
.	sentence-final punctuation

consists of a source sentence, a raw machine translation, a manually post-edited translation, and an automatically post-edited translation. The automatically post-edited translations have been produced by application of rule set I to the raw translations. Punctuation marks in the corpora are regarded as tokens and are thus separated from adjoining words with whitespace.

As mentioned, an important part of the method for generating rule sets is that all post-editing rules should be based on differences between raw machine translations and manually post-edited translations. Therefore, the raw machine translations and the manually post-edited translations of the medical generation corpus (as well as its source sentences) are stored in a file using the input data format of the rule-generation program (described in section 2.3.2).

In order to enable the generation of rules concerning certain parts of speech, the tokens of the medical generation corpus have been part-of-speech tagged using the HMM (hidden Markov model) tagger *HunPos*, version 1.0 (Halácsy et al., 2007). Each token is connected to its tag through a ¤ character. The tagging was performed using the default options of HunPos and a model for English trained on Wall Street Journal data from the Penn Treebank II and made available along with the tagger. A description of the Penn Treebank and its tag set is given by Marcus et al. (1993). Only the first two (or, in a few cases, three) characters of each tag are used in the generation corpus. The first two characters always provide the actual part-of-speech information, while the following characters (if any) keep either more detailed information (for example, if a noun is a proper or a common one) or no additional information at all (Marcus et al., 1993). The tags that will occur in this thesis are explained in table 3.1.

According to Halácsy et al. (2007), HunPos is capable of achieving a tagging accuracy of 96.58 % for English. However, the accuracy of the tagging of the generation corpus is probably lower. This is because differences between 'raw machine-translation English' and the English on which the tagger was trained have very likely caused a number of errors in the tagging of the raw machine

Figure 3.2: Four raw machine translations and corresponding manually post-edited translations from the medical generation corpus.

```
If¤IN this¤DT occasion¤NN is¤VB also¤RB missed¤VB ,¤, the¤DT
    student¤NN may¤MD hand¤VB in¤IN a¤DT written¤VB complementary¤JJ
    assignment¤NN .¤.
If¤IN this¤DT opportunity¤NN is¤VB also¤RB missed¤VB ,¤, the¤DT
    student¤NN may¤MD submit¤VB a¤DT written¤VB make-up¤JJ
    assignment¤NN .¤.

The¤DT nursing¤NN development¤NN to¤TO scientific¤JJ discipline¤NN
    is¤VB highlighted¤VB from¤IN both¤DT national¤JJ as¤IN
    international¤JJ perspective¤NN .¤.
The¤DT development¤NN of¤IN the¤DT nursing¤NN subject¤NN to¤TO a¤DT
    scientific¤JJ discipline¤NN is¤VB illustrated¤VB from¤IN
    a¤DT national¤JJ as¤RB well¤RB as¤IN an¤DT international¤JJ
    perspective¤NN .¤.

Being¤VB evil¤NN in¤IN the¤DT back¤RP an¤DT of¤IN the¤DT most¤RB
    common¤JJ reasons¤NN for¤IN disease¤NN in¤IN our¤PR$
    country¤NN .¤.
Pain¤NN in¤IN the¤DT back¤NN is¤VB one¤CD of¤IN the¤DT most¤RB
    common¤JJ causes¤NN of¤IN disease¤NN in¤IN our¤PR$
    country¤NN .¤.

The¤DT five¤CD ambulansentreprenör¤NN in¤IN Stockholm¤NN
    participate¤VB in¤IN the¤DT education¤NN .¤.
The¤DT five¤CD ambulance¤NN contractors¤NN of¤IN Stockholm¤NN
    participate¤VB in¤IN the¤DT education¤NN .¤.
```

translations. For example, some raw machine translations incorrectly include untranslated Swedish words. Figure 3.2 shows some raw machine translations and corresponding manually post-edited translations from the generation corpus.

3.3 Error Analysis

Evaluation metrics like BLEU, Meteor, and TER summarize large amounts of information about translation quality and translation errors into a single score. Sometimes this is an advantage, but in other cases, more detailed information can be very useful (apart from being interesting in its own right). Such a case is obviously manual development of post-editing rule sets, but automatic and semi-automatic development strategies might also benefit from a more detailed error analysis. For example, it might be very time-consuming to evaluate every post-editing rule that could possibly be generated from a parallel corpus. With the help of an analysis of the errors made by the basic translation system, the search for useful rules can instead be concentrated on (for example) words or parts of speech that are prone to translation errors.

Therefore, a three-part analysis of translation errors made by the Syllabus Translator was performed. Results from the analysis will be considered in

section 3.4, where some details of the method for generating rule sets will be elaborated, and in chapter 4, where two rule sets generated using the method will be described and evaluated. The analysed data were taken from the medical generation corpus, in order to leave the filter corpora and the test corpora unseen until later. An *error* is here defined to be a pair of differing chunks found by the rule-generation program (see section 2.3.2) when comparing raw machine translations to the corresponding manually post-edited translations. The chunks of an error will be referred to as the *incorrect chunk* and the *replacement chunk*, respectively. One of the chunks of a pair might be the empty string. Observe that the difference between an 'incorrect' chunk and the corresponding replacement chunk giving rise to an 'error' might be only in the tagging. This will be further discussed in section 3.3.3.

3.3.1 Chunk Lengths

In the first part of the analysis, the lengths of the incorrect chunks and the replacement chunks were studied. The rule-generation program was modified to extract all incorrect chunks and replacement chunks from the corpus. Then, information about the lengths of the chunks was gathered using shell commands and a Java program. It was found that 24,789 out of the 48,454 incorrect chunks are only one token long. Furthermore, 8,776 chunks are the empty string (indicating one or several missing tokens). The 39,678 non-empty incorrect chunks together comprise 66,619 tokens, or 1.68 tokens per chunk on average.

The replacement chunks were analysed in the same way. Here, the number of one-token chunks is 26,161 (out of totally 48,454, as before). There are also 5,244 cases in which the replacement chunk is the empty string (indicating that the corresponding incorrect chunk consisted of one or several superfluous tokens). The 43,210 non-empty replacement chunks together comprise 76,457 tokens, or 1.77 tokens per chunk on average. Thus, a majority of both the incorrect chunks and the replacement chunks are one token long, and on average, the non-empty incorrect chunks as well as the non-empty replacement chunks are less than two tokens long. Note that an incorrect one-token chunk might correspond to a replacement chunk of some other length, and vice versa. However, in 16,222 errors, both chunks are one-token chunks. This means that in 34,728 errors (71.7 % of the total number), at least one of the chunks is a one-token chunk.

3.3.2 Parts of Speech

In the second part of the analysis, parts of speech were in focus. As mentioned in section 3.2, the part-of-speech tagging most certainly contains errors. However, it is reasonable to assume that the tagging is correct enough to provide valuable information about what parts of speech are the most troublesome for the Syllabus Translator. Table 3.2 shows the most frequent combinations of complete sequences of part-of-speech tags in the chunks of the errors in the medical generation corpus. In all but one of the listed cases, each chunk contains at most one token (and therefore at most one tag). If a certain chunk contains no tokens, one or several tokens are missing or superfluous at the site in question. In the most frequent of the listed cases, an incorrect noun

Table 3.2: The most frequent (more than 400 occurrences) combinations of complete sequences of part-of-speech tags in the chunks of the errors in the medical generation corpus. EMPTY indicates that the chunk in question is the empty string (contains no tokens).

incorrect chunk	replacement chunk	frequency	relative frequency
NN	NN	5,968	12.3 %
IN	IN	2,730	5.6 %
EMPTY	,	2,419	5.0 %
VB	VB	2,247	4.6 %
EMPTY	DT	1,989	4.1 %
JJ	JJ	1,014	2.1 %
DT	EMPTY	780	1.6 %
NN	NN NN	503	1.0 %
NN	EMPTY	473	1.0 %
VB	EMPTY	435	0.9 %
EMPTY	NN	421	0.9 %
NN	VB	411	0.8 %

should be replaced by another noun. Corresponding cases for prepositions and subordinating conjunctions, verbs, and adjectives are also among the most frequent, as well as cases of a missing comma or determiner. Furthermore, the table includes cases where a determiner should be removed and where a noun should be replaced by two nouns. The relative frequencies are calculated in relation to the total number of errors (48,454, as before).

The incorrect chunks and the replacement chunks may also be studied separately. This results in the frequencies that are given in table 3.3 and table 3.4. The sequences of part-of-speech tags that were present in table 3.2 are also found here, along with a few others. Note that single commas and determiners are more common among the replacement chunks than among the incorrect chunks (where commas are not even common enough to be in the table). This points in the same direction as the findings of table 3.2, where missing commas and determiners are more common than superfluous determiners.

The chunks may also be split internally, so that the most frequent parts of speech for single tokens can be calculated. Such figures for the incorrect chunks and the replacement chunks are given in table 3.5 and table 3.6, respectively. The five most common parts of speech are the same for the two chunk types, but their order differs. The relative frequencies in the tables (computed in relation to the total number of tokens in the chunks in question) reveal that verbs and adjectives constitute larger parts of the incorrect chunks than of the replacement chunks, while the opposite is true for determiners and commas. A consequence of this is that adjectives and determiners shift ranks between incorrect chunks and replacement chunks. As regards the other listed parts of speech (nouns and prepositions and subordinating conjunctions), differences between the two chunk types are small. In the incorrect chunks, 85.8 % of all tokens belong to the five parts of speech in table 3.5. The corresponding figure for the replacement chunks and the six parts of speech in table 3.6 is 88.4 %. Note that the total number of tokens in the replacement chunks (76,457) is

Table 3.3: Complete sequences of part-of-speech tags with more than 700 occurrences in the incorrect chunks of the medical generation corpus. EMPTY indicates that the chunk in question is the empty string (contains no tokens).

incorrect chunk	frequency	relative frequency
NN	9,690	20.0 %
EMPTY	8,776	18.1 %
VB	4,326	8.9 %
IN	4,004	8.3 %
JJ	2,704	5.6 %
DT	1,282	2.6 %
JJ NN	802	1.7 %
NN NN	784	1.6 %
NN IN	733	1.5 %

Table 3.4: Complete sequences of part-of-speech tags with more than 700 occurrences in the replacement chunks of the medical generation corpus. EMPTY indicates that the chunk in question is the empty string (contains no tokens).

replacement chunk	frequency	relative frequency
NN	8,267	17.1 %
EMPTY	5,244	10.8 %
VB	4,118	8.5 %
IN	3,646	7.5 %
,	2,945	6.1 %
DT	2,632	5.4 %
JJ	2,086	4.3 %
NN NN	1,035	2.1 %
JJ NN	785	1.6 %
NN IN	730	1.5 %

much larger than the total number of tokens in the incorrect chunks (66,619). Thus, more tokens are missing than superfluous in the corpus.

The figures in the previous tables can also be compared to those in table 3.7 and table 3.8. The latter two tables show the most common parts of speech for all tokens in the raw translations and the manually post-edited translations, respectively. The raw translations consist of totally 314,470 tokens, while the manually post-edited translations consist of 324,308 (the same difference as between incorrect chunks and replacement chunks). Note that all parts of speech that have been mentioned as common in the analysed chunks are among the seven most common in the complete sets of raw translations and manually post-edited translations. That is, the top-ranking parts of speech in the incorrect chunks and the replacement chunks are common there partly because they are common in general. Nevertheless, some interesting tendencies can be traced in the tables.

For example, verbs as well as prepositions and subordinating conjunctions

Table 3.5: Part-of-speech tags with more than 2,000 occurrences in the incorrect chunks of the medical generation corpus.

tag	frequency	relative frequency
NN	22,523	33.8 %
VB	13,142	19.7 %
IN	10,188	15.3 %
JJ	7,258	10.9 %
DT	4,068	6.1 %

Table 3.6: Part-of-speech tags with more than 2,000 occurrences in the replacement chunks of the medical generation corpus.

tag	frequency	relative frequency
NN	24,914	32.6 %
VB	11,467	15.0 %
IN	11,265	14.7 %
DT	7,440	9.7 %
JJ	6,718	8.8 %
,	5,797	7.6 %

are relatively more common in both types of chunks than in the corresponding complete sets of translations. (Compare incorrect chunks to raw machine translations and replacement chunks to manually post-edited translations.) Therefore, prepositions and subordinating conjunctions rank third in the incorrect chunks, while they rank fourth in the complete raw machine translations. Adjectives, that rank third in the raw translations, seem to be a little less frequent in the incorrect chunks, something which might also be a factor behind the changed ranks. This tendency for adjectives to be less frequent in analysed chunks than in the corresponding complete translations is more apparent in the case of replacement chunks versus complete manually post-edited translations. This causes adjectives to rank lower than determiners in the replacement chunks, even though determiners are just a little more common there than in the manually post-edited translations as a whole. Finally, note that commas are more common in the replacement chunks than in the manually post-edited translations, where they, in turn, are more common than in the raw machine translations. The findings from the part-of-speech analysis will be discussed further in the next section.

3.3.3 Error Types

The aim of the third and final part of the analysis is to investigate the frequencies of different types of errors in the translations. For two reasons, the error-type analysis will only consider errors in which the incorrect chunk or the replacement chunk (or both) is a one-token chunk whose token belongs to one of the most common parts of speech (see section 3.3.2). The first reason is simply the high frequency of such errors, and thus, their importance. The second reason is that it is reasonable to believe that useful post-editing rules

Table 3.7: Part-of-speech tags with more than 7,000 occurrences in the raw machine translations of the medical generation corpus.

tag	frequency	relative frequency
NN	104,675	33.3 %
VB	40,734	13.0 %
JJ	36,961	11.8 %
IN	36,228	11.5 %
DT	26,285	8.4 %
CC	18,541	5.9 %
,	11,237	3.6 %
.	11,125	3.5 %

Table 3.8: Part-of-speech tags with more than 7,000 occurrences in the manually post-edited translations of the medical generation corpus.

tag	frequency	relative frequency
NN	107,066	33.0 %
VB	39,059	12.0 %
IN	37,305	11.5 %
JJ	36,421	11.2 %
DT	29,657	9.1 %
CC	18,919	5.8 %
,	16,424	5.1 %
.	11,286	3.5 %

can be built from many such errors (see section 3.4).

Using the rule-generation program, all possible rules (totally 48,454, just like the errors) were generated from the medical generation corpus. The context sizes for the rules were three tokens before and three tokens after every error (or less, in case the raw machine translation of the segment from which a rule was generated was not long enough to provide that much context). A shell script was used to select all those rules in which the incorrect chunk consisted of one noun, verb, preposition or subordinating conjunction, adjective, or determiner (see table 3.3), or in which the replacement chunk consisted of one noun, verb, preposition or subordinating conjunction, comma, determiner, or adjective (see table 3.4). The resulting rule set was sorted alphabetically, and duplicates were removed, so that only one copy of each rule remained. A Java program was then used to sample every 300th of the remaining 28,441 rules, resulting in a rule set comprising 94 rules.

These rules were manually categorized based on the types of errors that they correct. Some general principles were applied in this process. In order to avoid unnecessary constraints on the categorization process, the error categories were not predetermined. Instead, each analysed error that could not be fit into any of the already existing categories was allowed to give rise to a new category. Such a data-driven strategy could result in an inconsistent categorization with an unpractically large number of categories. Therefore, the categories were kept

rather broad, and the categorization was revised several times.

The categorization of each rule was based on what could be concluded from the rule itself. This means for example that even though a *superfluous word* rule (see below) might be part of a two-rule process moving a chunk to another position in a sentence (the other part being a *missing word* rule), this is not reflected in the categorization. In some cases, a rule was placed in several categories. However, each type of error was only counted once for each rule, even if it occurred more than once in the rule (for example, if there were several superfluous words). First, the error categories will be explained. Then, the results of the categorization will be commented upon.

- The category *wrong word* indicates that, according to the manually post-edited translation, (parts of) the incorrect chunk should be formulated differently.

- A rule is categorized as *superfluous word* if it is hard or impossible to find counterparts in the replacement chunk to one or several words in the incorrect chunk. In the extreme case, this means that the replacement chunk is the empty string. The opposite of this category is *missing word*. Here, one or several words in the replacement chunk have no counterpart in the incorrect chunk. One and the same rule may be categorized both as *wrong word* and as *superfluous word* or *missing word*, if it is obvious that one or several words are superfluous or missing in an incorrectly formulated chunk. The categories *superfluous punctuation mark* and *missing punctuation mark* are analogous to the word categories.

- The category *inflection error* is used for cases in which the system has found the correct word but used an incorrect form of it. That is, if the system has used an incorrect form of an incorrect word, only the *wrong word* category is used. Note that cases in which a word in an incorrect chunk is derived from a word in the corresponding replacement chunk (or vice versa) are categorized as *wrong word*, and not as *inflection error*.

- In case the incorrect chunk includes one or several Swedish words, the *word not translated* category is used.

- The category *casing error* is used for incorrect use of upper case or lower case letters (also when the casing error is caused by one or several superfluous or missing words in front of the miscased word).

- Finally, the category *no error* is used if the tag of a token differs between the incorrect chunk and the replacement chunk, while the token itself is identical in both chunks. This can happen because a token and its tag form a common string in the analysed corpus (see figure 3.2). The rule-generation program interprets such a string as a 'token' that can differ from another only in the tag part. Note that the existence of *no error* errors may have had a small impact on the results presented in section 3.3.1 and section 3.3.2.

The results of the categorization are given in table 3.9, and examples from the different categories in table 3.10. Because each rule might be placed into

Table 3.9: Categorization of 94 rules generated from the medical generation corpus, based on the types of errors that they correct. Some rules have been placed in several categories. The relative frequencies are calculated in relation to the number of analysed rules, and thus, their sum exceeds 100 %.

category	frequency	relative frequency
wrong word	40	42.6 %
superfluous word	20	21.3 %
inflection error	18	19.1 %
missing punctuation mark	13	13.8 %
missing word	10	10.6 %
word not translated	5	5.3 %
casing error	3	3.2 %
superfluous punctuation mark	1	1.1 %
no error	1	1.1 %

Table 3.10: Examples from the utilized error categories. Incorrect chunks are written in **boldface**. Incorrect chunks and replacement chunks that are the empty string are indicated by EMPTY. Two of the examples belong to more than one error category, because no pure examples from the categories in question were available.

category	incorrect chunk, including contexts	replacement chunk
wrong word	and different aspects **on** health from a	of
superfluous word	The course gives **an** advanced knowledge of	EMPTY
inflection error	able to define **problem** in a research	problems
missing punctuation mark	research and development **EMPTY** or to work	,
missing word	course participants and **EMPTY** course administration .	the
word not translated	of antibodies and **spädningstitrering** of antibodies .	dilution titration
casing error	The **nervous** system - from	Nervous
superfluous punctuation mark (plus superfluous word and word not translated)	, presbyopia , **pupillfunktion , test , the preconditions and** function of the	pupil
no error (plus missing word)	decrease the number **homeless** .	homeless people

several categories, the sum of the frequencies for all categories is 111, even though only 94 rules were analysed. The most frequent category is *wrong word*. This reflects a basic but difficult translation problem: choosing a correct and idiomatic translation of a word or group of words, given the context and the domain of the text. As described in section 2.3.1, the Syllabus Translator takes the context into account by using lexical transfer rules as a complement to dictionaries.

A common transfer-rule type concerns how to translate certain prepositions in certain lexical contexts (Pettersson, 2005, p. 14), something which can be a challenging problem for both humans and machine-translation systems. The categorized post-editing rules, as well as observations made by the developers of the Syllabus Translator (Pettersson and Sågvall Hein, 2011), indicate that much can still be done to improve the system's performance in this respect. To be more precise, a fourth of the *wrong word* errors (ten out of forty) concern incorrect preposition choices, even though prepositions and subordinating conjunctions make up only about a tenth of the corpus (see table 3.7 and table 3.8). Furthermore, table 3.2 shows that incorrect choice of preposition or subordinating conjunction ranks second when errors are categorized according to the parts of speech of their chunks.

The two most common parts of speech in the complete corpus, nouns and verbs (see table 3.7 and table 3.8), are about as frequent as prepositions in the chunks of the *wrong word* errors. Choosing correct translations can be a hard problem also for these parts of speech, but for different reasons. While prepositions constitute a closed class of words with a small set of members, the number of different nouns and verbs is large, and new ones are frequently introduced. Both these facts complicate the translation process and might cause *wrong word* and *word not translated* errors. Furthermore, the domain of the text might be a more important factor governing the correct choice of nouns and verbs than the immediate lexical context. As mentioned in section 2.3.1, the Syllabus Translator uses a dictionary hierarchy to take the domain of the text into account, but as could be expected, this strategy is not always successful.

Two important categories are those containing rules that handle *missing word* and *superfluous word* errors. As discussed above, some of these rules are not (or might at least not be) stand-alone rules. Instead, they are connected to other rules in processes that move chunks to new positions in sentences. That is, nothing is really missing or superfluous, but rather misplaced. Rules in the *superfluous word* and *missing word* categories might also be caused by too literal translations from Swedish. As shown in table 3.2, nouns and determiners are common in both these categories. Rules that concern determiners will be discussed in section 4.1.

The majority of the rules in the *missing punctuation mark* category insert commas into the text, something which is in accordance with the figures given in table 3.2. The need for more commas is also confirmed by the fact that commas are more frequent in the manually post-edited translations than in the raw machine translations (see table 3.7 and table 3.8).

About half of the rules in the third most frequent category, *inflection error*, address number problems in nouns. The category also includes some rules that handle such problems in verbs. Some Swedish nouns have identical singular and plural forms, and the same thing is true for all Swedish verbs. This is of course

a challenge for a system translating into English, where number distinctions are more common. Some other rules that concern verbs are also included.

3.4 Method Details

Section 3.5 will describe an implementation of the method for generating rule sets that was presented in section 3.1. However, five details pertaining to the method first have to be elaborated.

First, what rules should be included in the set of candidate rules to test against the filter corpus? It would of course be possible to let the set consist of *all* rules that could possibly be generated from a certain parallel corpus. However, testing only a subset of these rules gives more control over the composition of the final rule set and might also save time. Therefore, only *one-token rules* will be included in sets of candidate rules. A one-token rule is a rule that is based on an error in which the incorrect chunk or the replacement chunk (or both) is a one-token chunk. This is a somewhat arbitrary decision, but, like in section 3.3.3, two reasons for this restriction can be given.

The first reason is the high frequency of such errors, and thus, their importance (see section 3.3.1). The second reason is that it is not unreasonable to assume that errors whose differing chunks are short are caused by minor flaws in the basic translation system. Such flaws might be hard to correct but easy to compensate for with post-editing rules. On the other hand, errors whose differing chunks are longer might indicate more serious problems in the basic system that should rather be corrected than compensated for. The correct place for the limit between these two error categories is not necessarily the one chosen here. However, an implementation of the method that uses one-token rules can rather easily be adapted to handle rules based on errors with longer differing chunks, should this be needed.

Thus, each rule in a set of candidate rules is selected (included in the set) because it has a one-token incorrect chunk or a one-token replacement chunk. Like all other tokens in the generation corpus, the token of the chunk on which the selection is based has a part-of-speech tag. Before the selection, the user specifies which part-of-speech tags this token is allowed to belong to. This provides even more control over the composition of the final rule set. Note that if the tagging accuracy happens to be low (see section 3.2), this is not really a problem, as far as the rule-set quality is concerned. Only BLEU-increasing rules will be kept, so the worst possible consequence of a tagging error is that a rule concerning a token of an unintended part of speech is included in the final rule set.

Second, how large contexts before and after each error from the corpus should be used in the rules? In general, smaller contexts will result in higher recall (more matches) but lower precision when the rule is applied to a translation. Here, rules based on the same error but using different combinations of context sizes (from one token to maximum sizes given by the user) will be compared to each other, and only the one that increases the BLEU score the most will be kept. If none of the rules increases the score, all of them will of course be discarded. This flexible approach is a better choice than trying to fix a combination of context sizes for all rules, because the optimal sizes might be

different for different errors. Contexts of size zero will only arise when the raw machine translation of the segment from which a rule is generated does not provide any context. Certainly, an interesting variation would be to deliberately generate rules with such contexts, before or after the error. In the extreme case, both contexts would be of size zero, and the rule would prescribe that a chunk should always be changed into another. Such rules might be useful, but if the raw machine translations are really always incorrect in this respect, a better first try is probably to adjust the basic translation system (in particular, the dictionaries).

Third, how should casing differences be handled? The final rule set should preferably be useful also for post-editing of translations that differ in casing from the errors from which the rules were generated. Therefore, rules will be recased. That is, certain letters of the rules that have passed the filtering will be changed from upper case to lower case or vice versa, and the resulting rule set will also be filtered. The letters that might be changed (in different combinations) are the initial letters of the incorrect chunks, the replacement chunks, and the contexts before the errors.

Fourth, how should rules that have the same left-hand side (incorrect chunk and combination of contexts), but different right-hand sides (replacement chunks), be handled? Several rules in each such group might fulfil the criterion of raising the BLEU score of the raw machine translation of the filter corpus. The rules in each group will be compared to each other, and the best one (the one resulting in the highest BLEU score) will be kept.

Fifth, how should the rules in the rule set be ordered? Here, the order between rules will build upon the hypothesis that if the output (contexts and replacement chunk) of a certain rule is (partly) the input (contexts and incorrect chunk) of another rule, the former rule should be applied before the latter. Rules will be viewed as vertices in a directed graph, in which an edge from one vertex to another means that the output of the rule corresponding to the former vertex overlaps with the input of the rule corresponding to the latter. In order to simplify the sorting process, only exact matches between output and input will be considered. This means that a sequence of overlapping rules could actually be replaced by a new rule consisting of the left-hand side of the first rule in the sequence and the right-hand side of the last rule.

A directed graph will be constructed from the unordered rules, and then a so-called topological sort of the graph will be performed. Cormen et al. (2009, p. 603, 612–613) describe topological sorting in the following way. A topological sort of a directed graph is a linear ordering of all its vertices. This ordering must fulfil the requirement that if the graph contains an edge from vertex u to vertex v, then u will appear before v in the ordering. A topological sort can be found by means of a so-called depth-first search of the graph. The depth-first search assigns to each vertex a finishing time indicating when all its adjacent vertices have been examined completely. If the vertices are ordered according to decreasing finishing times, the result is a topological sort of the graph. Topological sorting requires the directed graph to be acyclic. In the case of a rule set, this means that there should be no cyclic dependencies between rules. An example of a cyclic dependency would be if the output of rule A were the input of rule B, the output of rule B were the input of rule C, and the output of rule C were the input of rule A. Unfortunately, cyclic dependencies

might occur in real rule sets. In case this happens, one of the edges of the corresponding graph (the last that has not yet been followed in the depth-first search) will simply not be considered. This will break the cycle and make the graph acyclic.

3.5 Implementation

The method for generating rule sets, including the details considered in the previous section, has been implemented as a shell script. The algorithm of the script is given in pseudocode in figure 3.3, under the name GENERATERULESET. The script makes use of Multi-BLEU, the rule-generation program, the rule-application program, and a program for topological sorting (see below). The implementation, as well as the method in general, could be improved in several ways. Their shortcomings are partly due to the fact that the script is wrapped around already existing programs, rather than being an integrated program. In chapter 5, possible enhancements of the method and the implementation will be discussed.

Using the argument names of figure 3.3, the input to the script consists of a tagged generation corpus *generation*, maximum context sizes *maxBefore* and *maxAfter* to use before and after errors in generated rules, sets *incorrectTags* and *replacementTags* of desired part-of-speech tags for the tokens of incorrect chunks and replacement chunks, a filter corpus *filter*, and a test corpus *test*. The script first calls Multi-BLEU to establish the BLEU scores *filterScore* and *testScore* of the raw machine translations of *filter* and *test*, when compared to the corresponding manually post-edited translations. The number of decimals in these and all other considered BLEU scores is four (given that the scores are in the range from 0 to 1), so score differences that are not large enough to be observable using four decimals will have no effect on the process. Then, for each possible combination of context sizes, the script calls the rule-generation program, asking it to generate a rule set A consisting of one rule for each error in *generation*. For each part-of-speech tag in *incorrectTags* and *replacementTags*, all matching one-token rules in A are selected and added to a separate rule set B. By 'matching' is meant that the token of the incorrect chunk or the replacement chunk (respectively) is tagged with the part-of-speech tag in question. The sets B together form the complete set of candidate rules to be filtered. Furthermore, all part-of-speech tags are removed from the selected rules. That is, the tags are only used in the selection of rules. When the rules are later applied, they have the format described in section 2.3.2.

Then, for each part-of-speech tag in *incorrectTags* and *replacementTags*, the candidate rules of all corresponding sets B are filtered using *filter*. The rules in the different rule sets B that are considered at the same time have been generated from the same error in *generation*, but use different combinations of context sizes. These rules form a group R and are compared to each other by being applied one by one to the raw machine translation of *filter* (using the rule-application program). The BLEU score of the resulting post-edited translation is established, and the rule that increases the BLEU score the most is selected and added to a rule set C. If several rules in R give the same BLEU score, the one that is evaluated first is selected. If no rule in R increases the

```
GENERATERULESET(generation, maxBefore, maxAfter, incorrectTags, replacementTags, filter, test)
 1  filterScore = BLEU score of the raw machine translation of filter
 2  testScore = BLEU score of the raw machine translation of test
 3  for beforeContext = 1 to maxBefore
 4      for afterContext = 1 to maxAfter
 5          A = the set of all rules that can be generated from generation
 6          for each part-of-speech tag in incorrectTags and replacementTags
 7              B = the set of all matching one-token rules in A
 8              remove all part-of-speech tags from the rules in B
 9  for each part-of-speech tag in incorrectTags and replacementTags
10      let C be an empty rule set
11      for each group R of rules generated from the same error (over all matching rule sets B)
12          bestScore = filterScore
13          for each rule r in R
14              editedFilter = the result of applying r to the raw machine translation of filter
15              ruleScore = BLEU score of editedFilter
16              if ruleScore > bestScore
17                  bestScore = ruleScore
18                  bestRule = r
19          add bestRule to C
20      remove duplicates from C
21      D = the set of all recased versions of the rules in C
22      remove duplicates from D
23      let E be an empty rule set
24      for each rule r in D
25          editedFilter = the result of applying r to the raw machine translation of filter
26          ruleScore = BLEU score of editedFilter
27          if ruleScore > filterScore
28              add r to E
29      F = a rule set consisting of all rules in C and E
30      remove duplicates from F
31  G = a rule set consisting of all rules in all rule sets F
32  remove duplicates from G
33  let H be an empty rule set
34  for each group R of rules in G that have the same left-hand side
35      bestScore = −1
36      for each rule r in R
37          editedFilter = the result of applying r to the raw machine translation of filter
38          ruleScore = BLEU score of editedFilter
39          if ruleScore > bestScore
40              bestScore = ruleScore
41              bestRule = r
42      add bestRule to H
43  topologically sort the rules of H
44  editedTest = the result of applying H to the raw machine translation of test
45  editedTestScore = BLEU score of editedTest
```

Figure 3.3: The algorithm of the script for generating rule sets.

score, no rule is selected.

Duplicates are removed from C so that only one copy of each rule remains. The reason that there might be duplicates of rules in C is that the same error, with the same combination of contexts, might have occurred several times in *generation*. Recased versions of the rules in C are then created and placed in a rule set *D*. Duplicates (that might result from the recasing) are removed from *D*, and the rules are filtered by being applied one by one to the raw machine translation of *filter*. All rules of *D* that increase the BLEU score are placed in a rule set *E*. Finally, the rule sets C and *E* are combined into a common rule set *F*, from which duplicates are removed.

Thus, the script has now produced a number of rule sets *F*, each one containing, for example, 'all good rules changing a noun into a chunk of tokens' or 'all good rules changing a chunk of tokens into a determiner'. These sets are combined to a set G, from which duplicates are removed. (Since the same rule might have been selected as a candidate rule both because of the tag of its incorrect chunk and because of the tag of its replacement chunk, the combined set might contain duplicates.) Then, the rules are grouped based on their left-hand sides, and in each group, the rule that results in the highest BLEU score when applied to the raw machine translation of *filter* is selected and added to a rule set *H*. If several rules in a group give the same BLEU score, the one that is evaluated first is selected. The rules of *H* are topologically sorted according to the criteria described in section 3.4. This is accomplished using a Java program that is called by the script. The program also performs a general reversal of the order of the rules. This is a consequence of the details of the sorting process, and will have no effect on the usefulness of the rule set. Finally, the set *H* is evaluated by being applied to the raw machine translation of *test*. The most interesting data produced by the script, apart from *H*, is the BLEU score *editedTestScore* of the resulting post-edited translation of *test*, and the BLEU score *testScore* of the raw machine translation of *test*. The difference between these two scores can be used to measure the quality of *H*. The script also outputs some other information about the set-generation process, such as the number of rules in the different produced rule sets.

4 Results

The previous chapter presented a script for generating sets of post-editing rules from two parallel corpora (one generation corpus and one filter corpus). In this chapter, two rule sets that have been generated using this script will be described and evaluated by being applied to two test corpora. These rule sets will be referred to as rule set II and rule set III. (Recall that rule set I is the manually developed rule set that was discussed in section 2.3.1.)

4.1 Rule Set II

In the generation of rule set II, the medical generation corpus and the medical filter corpus were used (see section 3.2). The maximum context sizes for the generated candidate rules were three tokens before and three tokens after each error in the generation corpus. As the minimum context sizes were one token before and one token after (see section 3.4), this means that there were nine (three to the power of two) different combinations of context sizes for preceding and following contexts. The decision not to include rules with larger contexts in the set of candidate rules is rather arbitrary. However, the larger the contexts of a rule, the more specific and the less generally useful it will be. This is once again the problem of precision and recall that was mentioned in section 3.4. As will be commented upon below, it was also found that among the rules that found their way into the final rule set, the most common combination of context sizes is one token before and one token after the errors. Furthermore, almost no rules in the set use contexts of three tokens. This suggests that testing rules with contexts of more than three tokens would, at best, have resulted in minor improvements of the final rule set.

Furthermore, only rules concerning the most common parts of speech in one-token incorrect chunks and one-token replacement chunks were considered. That is, the candidate rules were rules in which the incorrect chunk consisted of one noun, verb, preposition or subordinating conjunction, adjective, or determiner (see table 3.3), or in which the replacement chunk consisted of one noun, verb, preposition or subordinating conjunction, comma, determiner, or adjective (see table 3.4).

4.1.1 Composition

The complete rule set II consists of 1,011 rules. Information from the set-generation process gives interesting insights about the composition of this rule set. For each considered part of speech, table 4.1 and table 4.2 show the number of rules in different stages of the filtering process. The figures in

table 4.1 concern those rules that were included in sets *B* of candidate rules (see section 3.5) because of the parts of speech of the tokens in their incorrect chunks, while the ones in table 4.2 concern those rules that were included in sets of candidate rules because of the parts of speech of the tokens in their replacement chunks. Because nine different combinations of contexts sizes were tried, the number of non-recased *candidates* (candidate rules) for each part of speech is generally nine times the corresponding number of one-token chunks in the generation corpus, as presented in table 3.3 and table 3.4. Two very small discrepancies are due to the fact that the information was gathered in different ways. The *kept* non-recased rules are those that were included in a rule set C because they were capable of raising the BLEU score of the raw machine translation of the filter corpus, and the *unique* non-recased rules are those that remained when all duplicates had been removed. As for the recased rules, the tables show the number of (unique) *candidates* for each part of speech (rule sets *D*), as well as the number of rules that were *kept* (capable of raising the BLEU score and thus included in a rule set *E*) and the number of kept rules that were *new* (not present among the non-recased rules in the corresponding rule set C). Finally, the *sum* of the number of unique non-recased rules and the number of new recased rules is given (rule sets *F*).

Note that most of the candidate rules are rejected. The kept non-recased rules constitute 10.4 % of the candidate rules, if all rules generated from the same error in the generation corpus are counted as one, and only 1.2 % if each rule is counted separately. The removal of duplicates decreases the number of rules even more. This is true both inside each part-of-speech group (for example, note the reduction of the number of noun rules from 1,058 to 334 in table 4.1) and over all part-of-speech groups. To be more precise, 1,522 (722 plus 800) non-recased rules remained after the filtering and the removal of duplicates in each part-of-speech group, but this figure was reduced to 1,032 when the rules from all part-of-speech groups were combined and all duplicates were removed. This means that one and the same rule was rather often included in the set of candidate rules both because of the part of speech of the token in its incorrect chunk and because of the part of speech of the token in its replacement chunk.

Including recased rules, the number of finally approved rules over all part-of-speech groups from both tables was 1,533 (728 plus 805). This figure was reduced to 1,040 when duplicates were removed (from the combined rule set *G*), something which means that the recasing added only eight (1,040 minus 1,032) rules to the set of useful and unique rules. The selection of the best rule among several with the same left-hand side removed a further 29 rules before the final number of rules in rule set *H* (1,011) was reached. Apart from the general reversal of the order of the rules, only two rules were affected by the topological sorting. In fact, this means that these two rules were placed in the right order before the topological sorting.

A categorization of the rules in rule set II based on their context sizes can be found in table 4.3. The most frequent combination of context sizes (79.5 % of the rules) is one token before and one token after the errors that the rules correct. The dominance of this combination of context sizes might partly be an effect of the details of the filtering process. When evaluating a group of rules generated from the same error, the script always tests the rule with this

Table 4.1: Number of rules in different stages of the generation of rule set II. The rules in the table were selected as candidate rules because of the part-of-speech tags of the tokens in their incorrect chunks.

	non-recased rules			recased rules			
tag	candidates	kept	unique	candidates	kept	new	sum
NN	87,210	1,058	334	282	9	3	337
VB	38,934	391	144	134	0	0	144
IN	36,036	422	141	139	1	1	142
JJ	24,336	198	75	66	2	2	77
DT	11,538	82	28	28	0	0	28
all	198,054	2,151	722	649	12	6	728

Table 4.2: Number of rules in different stages of the generation of rule set II. The rules in the table were selected as candidate rules because of the part-of-speech tags of the tokens in their replacement chunks.

	non-recased rules			recased rules			
tag	candidates	kept	unique	candidates	kept	new	sum
NN	74,421	856	275	238	5	3	278
VB	37,062	461	133	124	0	0	133
IN	32,814	415	118	117	3	1	119
,	26,505	290	114	113	1	1	115
DT	23,688	453	114	106	2	0	114
JJ	18,783	130	46	41	0	0	46
all	213,273	2,605	800	739	11	5	805

combination first. Rules with larger combinations must beat the BLEU score of the rule that was tested first to be the chosen rule from the group. Even so, it is obvious that larger contexts are not of much help. The only important exception to this is that increasing the size of one of the contexts to two tokens apparently improves the result in some cases. For example, in a rule that substitutes 'cooperation' for 'collaboration' and where the context after the error is 'with', the context before the error should preferably be 'treatment in' instead of just 'in'. Rule set II also includes a fairly large number of rules in which one of the contexts is of size zero (because the generation corpus provided no context within the same segment), while the other context is of size one. All other evaluated combinations of context sizes are very rare, or even non-existing.

The rules in rule set II were also categorized based on the sizes of their incorrect chunks and replacement chunks. The result is presented in table 4.4. As could be expected from the rule selection criteria, at least one of the chunks is always exactly one token long. In more than half of the rules, both chunks have this size. When such a rule is applied, one token is simply replaced by another, and the number of tokens in the text being post-edited is not affected. Rules that increase the number of tokens are also common. This group is dominated by rules that replace a chunk of size zero with a chunk of size one,

Table 4.3: Categorization of all 1,011 rules in rule set II, based on the sizes of the contexts before and after the errors that they correct.

context sizes			
before	after	frequency	relative frequency
0	0	3	0.3 %
0	1	50	4.9 %
0	2	4	0.4 %
0	3	2	0.2 %
1	0	56	5.5 %
1	1	804	79.5 %
1	2	41	4.1 %
1	3	4	0.4 %
2	0	6	0.6 %
2	1	40	4.0 %
3	1	1	0.1 %

Table 4.4: Categorization of all 1,011 rules in rule set II, based on the sizes of their incorrect chunks and replacement chunks.

chunk sizes			
incorrect	replacement	frequency	relative frequency
0	1	244	24.1 %
1	0	20	2.0 %
1	1	523	51.7 %
1	2	114	11.3 %
1	3	29	2.9 %
1	4	9	0.9 %
1	5	6	0.6 %
1	6	2	0.2 %
1	7	5	0.5 %
1	8	1	0.1 %
1	14	1	0.1 %
1	26	1	0.1 %
2	1	42	4.2 %
3	1	10	1.0 %
4	1	4	0.4 %

but also includes many rules that replace a chunk of size one with a chunk of size two or three. Rules that cause larger increases in the number of tokens are also included in the rule set, but are fairly uncommon. Furthermore, there are quite a few rules that perform the opposite operation, that is, replace a chunk with a smaller one.

Out of the 1,011 rules in rule set II, 50 (every 20th) were chosen for a manual categorization. This categorization was made using the same error-type categories as in section 3.3.3. The result of the categorization is shown in table 4.5. Recall that the origin of the rules of rule set II is the complete

set of one-token rules (given certain part-of-speech restrictions) that could be generated from the medical generation corpus. In table 3.9, a sample of rules from this complete rule set is categorized. Therefore, the figures in table 4.5 and table 3.9 can be compared. However, comparisons should be done with same caution because of the small samples of rules. Such a comparison can be viewed in two ways. If the focus is placed on the *rules* of table 3.9, the comparison shows to what extent rules of different categories pass the filtering process, or, in other words, the general usefulness of different categories of candidate rules. On the other hand, focus can also be placed on the *errors* of table 3.9, if these are supposed to be representative of errors in translations produced by the Syllabus Translator. This means that the comparison shows to what extent the final rule set is capable of handling common translation errors.

The most frequent category in both table 4.5 and table 3.9 is *wrong word*. About half of the 23 rules in rule set II that belong to this category concern nouns in one way or the other, while preposition errors are addressed by about a fourth. Verb rules and adjective rules are also present. This is reminiscent of the make-up of the same category in the analysis in section 3.3.3.

The *missing word* category is notably more frequent in table 4.5 than in table 3.9, something which has an evident connection to the large number of rules that increase the number of tokens (see table 4.4). Most of the *missing word* rules in the sample from rule set II concern the insertion of determiners (definite and indefinite articles). This confirms an observation made in the development of the Syllabus Translator, namely that articles are often missing in translations produced by the system. The reason for this is probably a frequent (and correct) use of indefinite nominal phrases without articles in the Swedish source texts. This is reflected in the translations, although correct English translations of such phrases must often contain articles (Pettersson, 2005, p. 22).

Because 'the replacement chunk is a comma' is a criterion for the selection of rules to include in the set of candidate rules, it comes as no surprise that all the rules in the *missing punctuation mark* category concern commas. As mentioned, rule set II includes a notable number of rules replacing a chunk with a smaller one. This might be contrasted to the fact that the second most frequent category in table 3.9, *superfluous word*, is not at all that frequent in table 4.5. This might be due both to peculiarities of the categorized rules and to the fact that *superfluous word* rules and rules decreasing the number of tokens are not perfect counterparts (see the category definitions in section 3.3.3). Nevertheless, many of the candidate rules correcting *superfluous word* errors are apparently not generally applicable. The same thing can be said about the third most frequent category in table 3.9, *inflection error*, which is also less frequent in table 4.5.

4.1.2 Evaluation

When trying to improve a machine-translation system, it is necessary to have a baseline. That is, one has to know how good the translations of the unmodified system are. The impact of modifications to the system, such as a new set of post-editing rules, can then be measured through comparisons between the translation quality before and the translation quality after the modifications. Therefore, the evaluation metrics that were presented in section 2.1 (Multi-

Table 4.5: Categorization of 50 rules from rule set II, based on the types of errors that they correct. Some rules have been placed in several categories. The relative frequencies are calculated in relation to the number of analysed rules, and thus, their sum exceeds 100 %.

category	frequency	relative frequency
wrong word	23	46.0 %
missing word	13	26.0 %
missing punctuation mark	9	18.0 %
inflection error	3	6.0 %
word not translated	2	4.0 %
superfluous word	1	2.0 %
casing error	1	2.0 %

Table 4.6: BLEU, Meteor, and TER scores for the medical test corpus.

	Multi-BLEU	ZV-BLEU	Meteor	TER
no post-editing (raw machine translation)	0.5640	0.5719	0.4664	0.2526
rule set I (manually developed)	0.5778	0.5857	0.4715	0.2441
rule set II (automatically generated)	0.5876	0.5955	0.4736	0.2496
rule sets I and II	0.5958	0.6038	0.4768	0.2438
statistical post-editing	0.6653	0.6900	0.4976	0.1878

BLEU, ZV-BLEU, Meteor, and TER) were used to measure the quality of the raw machine translations of the two test corpora (the medical and the general) in relation to the manually post-edited translations. As will be evident, ZV-BLEU seems to consistently give higher scores than Multi-BLEU, something which is probably mainly due to that the ZV-BLEU scripts lowercase data before calculating the scores. Below, results of testing the statistical significance of differences in ZV-BLEU scores will often be given.

The raw machine translations of the two test corpora were then post-edited using the manually developed rule set I, and the quality of the resulting translations was measured using the evaluation metrics. As can be seen in table 4.6 and table 4.7, the metrics indicate that post-editing using rule set I improves the translation quality of both corpora. (Recall that a better translation is supposed to get a higher BLEU or Meteor score, but a lower TER score.) The difference in ZV-BLEU scores between raw machine translations and post-edited versions was found to be statistically significant for both corpora, given a significance level of 5 % and 10,000 bootstrap samples.

The 2,317 raw machine translations of the medical test corpus were also post-edited using the 1,011 rules in rule set II, resulting in totally 1,003 rule applications. Two obvious conclusions of these figures are that not all rules could be applied and that not all raw machine translations were modified. To be more exact, 458 different rules were applied. The most frequently used rule was applied 51 times, while 271 rules were applied only once. The rule applications increased the total size of the translations by 1.5 % in number of characters (from 251,668 to 255,457) and by 2.0 % in number of tokens (from 40,199 to 41,013). Thus, the large share of rules that increase the number of

tokens is evident in the result of the applications. Scores for the medical test corpus after it had been post-edited using rule set II can be found in table 4.6. Rule set II increases both versions of BLEU more than rule set I does. The difference in ZV-BLEU between the results of post-editing using the two rule sets is statistically significant, given the same conditions as before.

Recalling that the rules in rule set II were included in the set because of their ability to raise the (Multi-)BLEU score of a similar corpus (the medical filter corpus), it is not very surprising to find that the rule set also increases the BLEU score of the test corpus. Therefore, the additional opinions given by the Meteor and TER scores are very valuable. According to both Meteor and TER, rule set II improves the raw machine translations of the corpus. However, while the Meteor scores suggest that rule set II improves the raw machine translations more than rule set I does, the TER score indicates the opposite. Given that TER aims to measure the need for manual post-editing – a need which automatic post-editing is meant to reduce – the divergence of the TER score from the others is an interesting finding.

A combination of rule set I and rule set II was also evaluated on the medical test corpus. To be more precise, rule set II was applied to the output of the application of rule set I. In this case, 390 different rules from rule set II were applied totally 871 times. The maximum number of rule applications for a single rule was 51, while 228 rules were only applied once each. The rule applications increased the total size of the translations by 1.4 % in number of characters (from 251,424 to 254,976) and by 1.9 % in number of tokens (from 40,129 to 40,901). All metrics agree that the combination of rule set I and rule set II outperforms both the single rule sets. The differences in ZV-BLEU scores are statistically significant. This indicates that it is possible to identify different translation errors using manual and automatic methods.

As described in section 2.2.2, several researchers have found that statistical machine-translation systems can successfully be used for automatic post-editing. Therefore, the results of post-editing using rule set II were compared to results of post-editing using the statistical machine-translation system Moses (Koehn et al., 2007). The system was trained to translate from 'raw machine-translation English' to 'manually post-edited English'. Two corpora were used in the training. The first was a version of the medical generation corpus that had been cleaned using a script included in Moses and then lowercased, while the second was a version of the medical filter corpus that had been lowercased. Among other adjustments, the cleaning script removes lines that exceed a certain length threshold (in this case, 100 tokens). The cleaning reduced the number of segments in the generation corpus from 18,540 to 18,500. The training was performed in three steps. First, a 5-gram language model of 'manually post-edited English' was built from the generation corpus using the SRILM toolkit (Stolcke, 2002; Stolcke et al., 2011). When a language model is built, probability estimates for n-grams that have few or no occurrences in the training corpus are often improved through *smoothing* (Jurafsky and Martin, 2009, p. 131–145). In this case, smoothing in the shape of so-called modified Kneser–Ney discounting (Chen and Goodman, 1998) for 1-grams, 2-grams, and 3-grams was used. Second, a translation model was built from the same corpus using the default settings of Moses. Third, the translation model was tuned on the filter corpus.

Table 4.7: BLEU, Meteor, and TER scores for the general test corpus.

	Multi-BLEU	ZV-BLEU	Meteor	TER
no post-editing (raw machine translation)	0.6217	0.6330	0.4745	0.2487
rule set I (manually developed)	0.6354	0.6465	0.4808	0.2409
rule set II (automatically generated)	0.6137	0.6214	0.4719	0.2568
rule sets I and II	0.6259	0.6334	0.4772	0.2493
rule set III (automatically generated)	0.6259	0.6372	0.4756	0.2467
rule sets I and III	0.6376	0.6488	0.4813	0.2395
statistical post-editing	0.5312	0.5713	0.4304	0.2824

A lowercased version of the medical test corpus was post-edited using the trained system. The resulting translation was recased (in order to restore the original casing) using the recaser of Moses, trained on the cleaned manually post-edited translation of the medical generation corpus. The evaluation metrics agree that the statistically post-edited version of the medical test corpus is of higher quality than all other translation versions, as can be seen in table 4.6. The difference in ZV-BLEU between the statistically post-edited version and the version post-edited using both rule set I and rule set II is statistically significant.

Rule set II was also applied to the 3,755 raw machine translations of the general test corpus. This resulted in totally 906 applications of 223 different rules. That is, even though this corpus is larger than the medical test corpus, fewer rules could be applied to it. The total size of the translations was increased by 0.8 % in number of characters (from 368,194 to 371,032) and by 1.1 % in number of tokens (from 58,002 to 58,614). As shown in table 4.7, all evaluation metrics indicate that the application of rule set II had a negative impact on the translation quality of the general test corpus. The difference in ZV-BLEU scores between the raw machine translations and the version post-edited using rule set II is statistically significant. Rule set II was also used to post-edit a version of the general test corpus that had already been post-edited using rule set I. Here, 201 different rules from rule set II were applied totally 805 times. The increase in size was 0.7 % in number of characters (from 368,305 to 370,978) and 0.9 % in number of tokens (from 58,051 to 58,567). The resulting version was of significantly lower quality than the one that had only been post-edited using rule set I.

The statistical machine-translation system that had been trained on the medical generation corpus and tuned on the medical filter corpus was also used to post-edit a lowercased version of the raw machine translations of the general test corpus. The resulting translation was then recased using the recaser that had been trained on the medical generation corpus. As shown in table 4.7, the evaluation metrics indicate that this deteriorated translation quality even more than the post-editing using rule set II. The difference in ZV-BLEU scores between these two post-edited versions is statistically significant. It seems reasonable to assume that the negative impact of rule set II and the statistical system is caused by the fact that they are based on medical corpora. Apparently, both rule set II and the statistical system are too domain-specific to

Table 4.8: Number of rules in different stages of the generation of rule set III. The rules in the table were selected as candidate rules because of the part-of-speech tags of the tokens in their incorrect chunks.

	non-recased rules			recased rules			
tag	candidates	kept	unique	candidates	kept	new	sum
NN	87,210	111	25	22	0	0	25
VB	38,934	10	2	2	0	0	2
IN	36,036	31	8	8	0	0	8
JJ	24,336	34	3	3	0	0	3
DT	11,538	0	0	0	0	0	0
all	198,054	186	38	35	0	0	38

Table 4.9: Number of rules in different stages of the generation of rule set III. The rules in the table were selected as candidate rules because of the part-of-speech tags of the tokens in their replacement chunks.

	non-recased rules			recased rules			
tag	candidates	kept	unique	candidates	kept	new	sum
NN	74,421	94	20	19	0	0	20
VB	37,062	18	6	5	0	0	6
IN	32,814	50	8	8	0	0	8
,	26,505	7	2	2	0	0	2
DT	23,688	25	2	2	0	0	2
JJ	18,783	0	0	0	0	0	0
all	213,273	194	38	36	0	0	38

be successfully applied to raw machine translations from a broader spectrum of domains (scientific fields). Very likely, the problem is caused by differences in vocabulary between the medical and the general corpora.

4.2 Rule Set III

The finding that rule set II does not improve the translation quality of the general test corpus raises the question if the *method* for generating rule sets is general enough to produce a domain-adapted rule set from the same (medical) generation corpus. That is, is it possible to perform domain adaptation by keeping the generation corpus, but replacing the filter corpus with one that is based on material from the desired (general) domain? In order to investigate this, another rule set, called rule set III, was generated using the script, the medical generation corpus and the general filter corpus. The same context sizes and parts of speech were used as in the generation of rule set II.

4.2.1 Composition

Only 46 rules passed the filtering and were finally included in rule set III. Table 4.8 and table 4.9 correspond to table 4.1 and table 4.2. As could be

Table 4.10: Categorization of all 46 rules in rule set III, based on the sizes of the contexts before and after the errors that they correct.

context sizes			
before	after	frequency	relative frequency
0	0	1	2.2 %
0	1	3	6.5 %
1	0	8	17.4 %
1	1	31	67.4 %
1	2	1	2.2 %
2	0	1	2.2 %
2	1	1	2.2 %

Table 4.11: Categorization of all 46 rules in rule set III, based on the sizes of their incorrect chunks and replacement chunks.

chunk sizes			
incorrect	replacement	frequency	relative frequency
0	1	3	6.5 %
1	1	32	69.6 %
1	2	8	17.4 %
1	4	1	2.2 %
2	1	2	4.3 %

expected from the small number of rules in the complete rule set, very few of the non-recased candidate rules were found to be useful. The kept non-recased rules constitute 0.83 % of the candidate rules, if all rules generated from the same error in the generation corpus are counted as one, and only 0.092 % if each rule is counted separately. As in the case of rule set II, the number of rules is heavily reduced when duplicates are removed. Clearing each part-of-speech group from duplicates reduces the 380 (186 plus 194) kept rules to 76 (38 plus 38). Combining the rules from all part-of-speech groups and removing duplicates results in the final set of 46 rules. Note that none of the recased candidate rules was kept. No rules were removed by the process that compares rules with identical left-hand sides, and apart from the general reversal of the order of the rules, no rules were reordered by the topological sorting.

The most frequent combination of context sizes is, like in rule set II, one token before and one token after the errors that the rules correct (see table 4.10). Another fairly common combination is one token before and no token after the errors. Like in rule set II, both the incorrect chunk and the replacement chunk are of size one in many of the rules (see table 4.11). Rules that increase the number of tokens are also common.

As in the sample from rule set II, the most frequent error category among the rules in rule set III is *wrong word* (see table 4.12). Rules that concern nouns account for almost half of the rules in this category, and the same thing is true for rules that concern prepositions. The second most frequent category, *inflection error*, almost exclusively deals with number errors in nouns, while

Table 4.12: Categorization of all 46 rules in rule set III, based on the types of errors that they correct. Some rules have been placed in several categories. The relative frequencies are calculated in relation to the number of analysed rules, and thus, their sum exceeds 100 %.

category	frequency	relative frequency
wrong word	22	47.8 %
inflection error	14	30.4 %
missing word	9	19.6 %
no error	3	6.5 %
missing punctuation mark	2	4.3 %
casing error	2	4.3 %
word not translated	1	2.2 %
superfluous word	1	2.2 %

almost all *missing word* rules add determiners. Thus, the rules in these categories seem to correct similar errors as those in the same categories in rule set II (see table 4.5). However, the frequencies of some categories differ. Rules that correct inflection errors are more frequent in rule set III than in rule set II, while rules that concern missing words and, especially, missing punctuation marks are less frequent. In comparison to the sample from the generation corpus (table 3.9), *inflection error* and *missing word* rules are more frequent, but for *superfluous word* and *missing punctuation mark*, the opposite is true. As mentioned before, all comparisons between different rule sets should be done with some caution because of the small number of studied rules.

4.2.2 Evaluation

According to all four evaluation metrics, rule set III is capable of improving the raw machine translation of the general test corpus, even though the improvements are smaller than when rule set I is used to post-edit the corpus (see table 4.7). The difference in ZV-BLEU scores between the raw machine translation and the version post-edited using rule set III is statistically significant, as well as the difference between the rule set III version and the rule set I version. Also, the improvements caused by rule set III are modest when compared to those achieved when rule set II is applied to the medical test corpus. This is of course not surprising, given the small number of rules in rule set III. Out of the 46 rules in the set, 40 could be applied to the test corpus. The total number of applications of these rules was 275, and the most frequent rule was applied 40 times. The total size of the translations increased by 0.2 % in number of tokens (from 58,002 to 58,091) and by 0.2 % in number of characters (from 368,194 to 368,795).

Post-editing using the combination of rule set I and rule set III yields better results than just using rule set I. The difference in ZV-BLEU scores is statistically significant. Here, 31 rules from rule set III were applied totally 186 times to the output of the application of rule set I. The highest number of applications for a single rule was 30. There total size of the translations increased a little – by 0.04 % in number of tokens (from 58,051 to 58,072) and by 0.1 % in number of characters (from 368,305 to 368,380).

5 Discussion

The purpose of this study has been to investigate if it is possible to use parallel corpora to automatically generate sets of post-editing rules that are capable of increasing the quality of machine translations. As shown in the previous chapter, this is indeed possible using a method based on generation of a set of candidate rules from one corpus and filtering of this set through another. In this concluding chapter, the method and its implementation will be discussed and suggestions about possible enhancements will be given.

5.1 Method

Better post-editing results than the ones presented in chapter 4 can very likely be achieved within the rule-based post-editing paradigm. The most obvious way to develop the method for generating rule sets is perhaps to include more rules in the set of candidate rules. As discussed in section 3.4, restrictions on this set regarding for example chunk lengths and parts of speech have certain advantages. However, nothing precludes the possibility of relaxing the restrictions that have been used here, or even removing them totally. Another way of developing the method would be to use other types of candidate-rule restrictions. For example, it would be interesting to investigate the effect of simultaneously imposing part-of-speech restrictions on both incorrect chunks and replacement chunks, of using restrictions based on other features than part of speech, or of imposing restrictions on contexts of errors.

Furthermore, more complex rule types could be used. Part-of-speech tags for tokens in contexts and incorrect chunks could be included in the post-editing rules. (Recall that in this study, tags were only used in the selection of candidate rules.) Rules could also make use of, for example, long-distance dependencies (and not only immediate contexts), wildcards (and not only fixed tokens in contexts and chunks), and contexts that do *not* include certain tokens. Another idea is that automatically generated rule sets might be improved by manual adjustments, such as editing of rules, addition of new rules inspired by the automatically generated ones, or even removal of rules that seem strange.

The recasing and sorting of rules had only a small effect on rule set I and did not change rule set II at all. This could be seen as a reason to remove these steps from the set-generation process (in order to make it more efficient), but it might also be possible to improve them and make them more useful. More rules (for example, all candidate rules, and not only those increasing the BLEU score of the filter corpus) could be recased in more ways, or other strategies for handling casing differences could be tried. The sorting could be based on more

relations between rules than exact matches between outputs and inputs.

In section 4.1.2, a statistical machine-translation system that had been trained to perform post-editing was evaluated. The results (when keeping to the domain of the training data) were much better than those that were achieved through post-editing using rule set I and rule set II. This could be seen as a reason to use a statistical post-editing strategy instead of a rule-based one. Note, however, that the (almost) complete medical generation corpus was used in the training of the statistical system, while only some of the rules that could be generated from this corpus were used as candidate rules in the generation of rule set II. Furthermore, different amounts of training data might be needed to produce a statistical system and a rule set capable of achieving the same translation quality. Therefore, a comparison between the rule sets and the statistical system might be unfair in some direction.

Note also that even though rule set II decreases the quality of the translations of the general test corpus, this decrease is much smaller than the one caused by the statistical system. This indicates that the rule-based strategy is not as domain-dependent as the statistical one. Furthermore, the experiment in domain-adaptation that resulted in rule set III could be reversed, so that a general generation corpus and a more specific filter corpus would be used. This might result in a better (larger) rule set than rule set III. On the other hand, the statistical system might also be domain-adapted and improved in other ways through the use of other corpora for model building and tuning. As a final remark on rule-based versus statistical post-editing, note that rule-based post-editing has the general advantage that rules can easily be interpreted and adjusted by humans, something which can be very valuable. In contrast, the inner workings of statistical systems are not as easy to adjust.

5.2 Implementation

The shell-script implementation that was described in section 3.5 is not very efficient, and could be improved in several ways. The most important improvement would probably be to integrate all parts of the process into a complete program for rule-set generation, instead of relying on a combination of separate programs. This would make it possible to use data from the different parts in a much more efficient way. For example, even if a certain error in the generation corpus does not fulfil desired restrictions, the current script first generates a rule from it and then discards the rule. This should of course be avoided. An integrated program would also make it easier to gather information about the process.

Another source of inefficiency is that the same rule might be evaluated several times. There are two reasons for this. First, rules are initially generated from all errors in the generation corpus. This means that identical rules might be generated from different errors (locations) in the corpus. Such duplicates should preferably be removed before the filtering process starts, but this is currently not done. Second, a rule generated from the same error might be included in the set of candidate rules twice, both because of the part-of-speech tag of the token in its incorrect chunk and because of the tag of the token in its replacement chunk. Improved candidate-rule restrictions (see above) and, once

again, removal of duplicates would be ways to avoid this problem.

There would also be a gain in efficiency if the BLEU score for the filter corpus were not always totally recomputed, but rather just updated based on the changes caused by the application of each rule. If a rule cannot be applied at all, the score computation could of course be avoided altogether. Scores could also be saved and reused, for example in comparisons between rules with the same left-hand side. Finally, it might be noted that the script frequently stores temporary results to disk. This is a time-consuming process that should of course be avoided as much as possible.

In short, there are a large number of ways in which the automatic generation of post-editing rule sets could be improved. Even so, this study shows that useful rule sets can be generated from parallel corpora of raw machine translations and manually post-edited translations. The more the Syllabus Translator and other machine-translation systems are used, the more corpus data will be available, and the better results can be expected from systems for automatic post-editing that make use of such data.

A Cleaning and Normalization

This appendix will summarize the cleaning and normalization measures that were taken in the process of building the corpora that have been used in this study. One effect of these measures is that the total number of segments is lower in the corpora than in the translation memories from which they were built. To be exact, the translation memories comprise 27,062 'medical' segments and 12,507 'general' segments, while the corresponding corpora comprise 23,174 and 7,511 segments, respectively (see section 3.2).

Segments lacking source sentences or manually post-edited translations were removed. So were also segments with source sentences containing newline characters, as these were found to cause problems later in the process. In the manually post-edited translations, newline characters were replaced by spaces. A number of segments were found to have English source sentences. A possible reason for the presence of English source sentences is that a text that is mainly written in Swedish might very well contain a few sentences in English. These segments were not removed. It is likely that the Syllabus Translator will encounter English source sentences again, and it is therefore useful to have post-editing rules for errors that arise when the system tries to translate such sentences. (As mentioned in section 2.3.1, all source sentences are analysed as if they were written in Swedish.) In case a segment contained several manually post-edited translations, the most recent one was always chosen.

Raw machine translations were not generally stored in the translation memories. Therefore, raw machine translations and automatically post-edited translations based on these were produced through re-translation of the source sentences. Note that an important consequence of this is that within a segment, there is only an indirect connection between the raw machine translation and the automatically post-edited translation, on the one hand, and the manually post-edited translation, on the other. To be more precise, no manually post-edited translation is really based on the corresponding raw machine translation or automatically post-edited translation. Rather, each manually post-edited translation is based on the automatically post-edited translation that was once presented to the human post-editor. This latter translation might have been identical to the raw re-translation or the automatically post-edited re-translation (or both), but because the Syllabus Translator is constantly updated (Pettersson, 2011), it might also have been different from them. This somewhat uncertain situation is of course not optimal. However, what is more important to this study is that the manually post-edited translations have been considered correct by a human post-editor and can thus be used as reference translations. For unknown reasons, the re-translation of some source sentences did not result in any real translations, but in empty strings. Segments where this happened were

Table A.1: Replacements that were made in the building of the corpora.

original					replacement	
ref I	ref II	ref III	code	name	char(s)	code(s)
	&#34;			quotation mark	"	34
&		&	38	ampersand	and	97 110 100
'	&#39;			apostrophe	'	39
,				comma	,	44
<	&#60;	<		less-than sign	<	60
>	&#62;	>		greater-than sign	>	62
			145	left single quotation mark	'	39
’			146	right single quotation mark	'	39
“			147	left double quotation mark	"	34
”	&#148;		148	right double quotation mark	"	34
•			149	bullet	·	183
–	&#150;		150	en dash	--	45 45
	&#8203;			zero width space		

removed from the medical corpora, but not from the general.

In order to make the post-editing rules that were to be generated from the corpora more generally applicable and to avoid potential problems related to inconsistencies in the character encoding, all data in the corpora were converted to the standard character encoding of the Convertus systems, *ISO-8859-1*. Characters not present in this encoding were replaced by close counterparts. For similar reasons, ampersands and character references (Raggett et al., 1999, chapter 24.2) were replaced and markup was removed. Table A.1 summarizes the replacements that were made. The left part of the table (*original*) contains replaced character references and character codes. The character references were of three different types, each of which has its own column in the table (*ref I*, *ref II*, and *ref III*). The character names that are given in the table are taken from the Unicode Character Name Index (The Unicode Consortium, 2012). The correspondence between a certain character reference or character code and a certain character (name) are based upon the informed guess that the character encodings used for the original texts were ISO-8859-1 (Whistler, 1999) and *windows-1252* (Steele, 1998). The right part of the table contains the *replacement* characters and the corresponding decimal character codes in ISO-8859-1. When necessary (in order not to make 'R&D' read 'RandD'), a space (character code 32) was inserted before and after the 'and' replacing the character with code 38 (the ampersand). Zero width spaces were replaced by nothing.

Bibliography

Jeffrey Allen and Christopher Hogan. Toward the Development of a Postediting Module for Raw Machine Translation Output: a Controlled Language Perspective. In *Third International Controlled Language Applications Workshop (CLAW2000)*, 2000.

Satanjeev Banerjee and Alon Lavie. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005.

Björn Beskow. *Unification-Based Transfer in Machine Translation*. Number 24 in RUUL (Reports from Uppsala University, Department of Linguistics). Uppsala University, Department of Linguistics, 1993.

Eric Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21:543–565, 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 EACL*, pages 249–256, 2006.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (Meta-) Evaluation of Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, 2007.

Stanley F. Chen and Joshua Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2009.

Michael Denkowski and Alon Lavie. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*, 2011.

Mark-Jason Dominus, Ned Konz, and Tye McQueen. *Man page of Algorithm::Diff (Diff.pm)*, 2004.

Loïc Dugast, Jean Senellart, and Philipp Koehn. Statistical Post-Editing on SYSTRAN's Rule-Based Translation System. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 220–223, 2007.

Jakob Elming. Transformation-based correction of rule-based MT. In *EAMT-2006: 11th Annual Conference of the European Association for Machine Translation*, pages 219–226, 2006.

Rafael Guzmán. Automating MT post-editing using regular expressions. *Multilingual*, 18(6):49–52, September 2007.

Péter Halácsy, András Kornai, and Csaba Oravecz. HunPos – an open source trigram tagger. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 209–212, 2007.

James W. Hunt and Thomas G. Szymanski. A fast algorithm for computing longest common subsequences. *Communications of the ACM*, 20(5):350–353, May 1977.

Pierre Isabelle, Cyril Goutte, and Michel Simard. Domain adaptation of MT systems through automatic post-editing. In *Machine Translation Summit XI*, pages 255–261, 2007.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing – An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Education, 2009.

Kevin Knight and Ishwar Chander. Automated Postediting of Documents. In *12th National conference of the American Association for Artificial Intelligence (AAAI 1994)*, 1994.

Philipp Koehn. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, 2004.

Philipp Koehn and Christof Monz. Manual and Automatic Evaluation of Machine Translation between European Languages. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 102–121, 2006.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180, 2007.

A.-L. Lagarda, V. Alabau, F. Casacuberta, R. Silva, and E. Díaz-de-Liaño. Statistical Post-Editing of a Rule-Based Machine Translation System. In *Proceedings of NAACL HLT 2009: Short Papers*, pages 217–220, 2009.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

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)*, pages 311–318, 2002.

Eva Pettersson. Pilotstudie om maskinöversättning inom ramen för Projekt Kursdatabas - Utveckling av språkliga resurser för ett vetenskapsområde samt utvärdering. Technical report, Uppsala University, Department of Linguistics and Philology, 2005.

Eva Pettersson. *Kursplaneöversättaren – Ett automatiskt översättningsstöd för översättning av akademiska kursplaner från svenska till engelska*, 2010.

Eva Pettersson. Personal communication, 2011.

Eva Pettersson and Anna Sågvall Hein. Personal communication, 2011.

Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.01 Specification, 1999. URL `www.w3.org/TR/html4/sgml/entities.html`. Visited 20th April 2012.

Lennart Sandberg and Richard Sandberg. *Elementär statistik*. Studentlitteratur, 2002.

Michel Simard, Cyril Goutte, and Pierre Isabelle. Statistical Phrase-based Post-editing. In *Proceedings of NAACL HLT 2007*, pages 508–515, 2007a.

Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. Rule-based Translation With Statistical Phrase-based Post-editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206, 2007b.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, 2006.

Shawn Steele. cp1252 to unicode table, 1998. URL `www.unicode.org/Public/MAPPINGS/VENDORS/MICSFT/WINDOWS/CP1252.TXT`. Visited 20th April 2012.

Andreas Stolcke. SRILM – An Extensible Language Modeling Toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 901–904, 2002.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. SRILM at Sixteen: Update and Outlook. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.

Sara Stymne and Lars Ahrenberg. Using a Grammar Checker for Evaluation and Postprocessing of Statistical Machine Translation. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*, pages 2175–2181, 2010.

Anna Sågvall Hein. *Maskinöversättning idag*. Uppsala University, Department of Linguistics and Philology, 2008.

Anna Sågvall Hein. Personal communication, 2012.

Anna Sågvall Hein. Preferences and Linguistic Choices in the Multra Machine Translation System. In Robert Eklund, editor, *NODALIDA '93 – Proceedings of '9:e Nordiska Datalingvistikdagarna'*, 1994.

Anna Sågvall Hein. Language Control and Machine Translation. In *Proceedings of the 7th International Conference on Theoretical and Methodological Issues in Machine Translation*, 1997.

Anna Sågvall Hein, Eva Forsbom, Jörg Tiedemann, Per Weijnitz, Ingrid Almqvist, Leif-Jöran Olsson, and Sten Thaning. Scaling Up an MT Prototype for Industrial Use - Databases and Data Flow. In *Proceedings of LREC 2002 – Third International Conference on Language Resources and Evaluation*, volume 5, pages 1759–1766, 2002.

Anna Sågvall Hein, Eva Forsbom, Per Weijnitz, Ebba Gustavii, and Jörg Tiedemann. MATS - A Glass Box Machine Translation System. In *Proceedings of the Ninth Machine Translation Summit*, pages 491–493, 2003.

The Unicode Consortium. Unicode Character Name Index, 2012. URL `www.unicode.org/charts/charindex.html`. Visited 20th April 2012.

Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl*. O'Reilly & Associates, second edition, 1996.

Per Weijnitz. String based post editing. Technical report, Convertus AB, 2010.

Per Weijnitz, Anna Sågvall Hein, Eva Forsbom, Ebba Gustavii, Eva Pettersson, and Jörg Tiedemann. The machine translation system MATS - past, present & future. In *Proceedings of RASMAT'04 (Recent Advances in Scandinavian Machine Translation)*, 2004.

Ken Whistler. ISO/IEC 8859-1:1998 to Unicode, 1999. URL `www.unicode.org/Public/MAPPINGS/ISO8859/8859-1.TXT`. Visited 20th April 2012.

Ying Zhang and Stephan Vogel. Measuring Confidence Intervals for the Machine Translation Evaluation Metrics. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2004)*, 2004.