Karlsruhe Institute of Technology

Source Sentence Reordering for English to Japanese Machine Translation

Bachelor Thesis of

Timo Abele

At the Department of Informatics
Institute for Anthropomatics and Robotics (IAR)

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Dr. Sebastian Stüker
Advisor:	Dr. Jan Niehues
Second advisor:	Teresa Herrmann, M. Sc.

Duration: 5. July 2014 – 4. November 2014

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 4.11.2014

. . . .

(**Timo Abele**)

Abstract

Maschinelle Übersetzung, d.h das Übersetzen von Text von einer Sprache in eine andere, ist in den letzten Jahren zu einer wichtigen Bereicherung unseres Alltags geworden. Trotz gewisser Unzulänglichkeiten oder manchmal unnatürlicher Übersetzungen, erfreut sich maschinelle Übersetzung Beliebtheit, wenn menschliche Übersetzer zu teuer sind. Während das Übersetzen von Wörtern einfach mit einem Lexikon bewerkstelligt werden kann, bleibt reordering (der Platzwechsel der einzelnen Wörter) eine große Herausforderung in der maschinellen Übersetzung. Besonders phrasenbasierte Ansätze haben hier Probleme, da sie nicht von sich aus mit syntaktischer Information arbeiten.

Ein neuerlicher Ansatz, dieses Problem in der maschinellen Übersetzung anzugehen, ist es, solche *reordering*-Regeln automatisch zu erlernen und vor der eigentlichen Übersetzung auf die Quellsätze anzuwenden. Dieser Ansatz ist kürzlich durch ein verfeinertes Reorderingmodell erweitert worden, das die Informationen aus Syntaxbäumen nutzt, um mögliche Umordnungen zu bestimmen. Dieses Modell ist erfolgreich auf zwei europäischen Sprachpaaren getestet worden. In dieser Thesis wenden wir diesen Ansatz auf ein sehr gegensätzliches Sprachpaar an: Englisch und Japanisch. Japanisch ist eine Subjekt-Objekt-Verb (SOV) Sprache, d.h. das Subjekt steht an vorderster Stelle, gefolgt vom Objekt und am Ende steht das Verb. OSV Sätze sind auch möglich, es ist aber zwingend, dass das Verb an letzter Stelle steht. Dies ist ein Leichtes für ein regelbasiertes System, aber in der statistischen maschinellen Übersetzung müssen wir ein Modell entwickeln. Wir vergleichen wortklassenbasierte Regeln mit Regeln die auf Syntaxbäumen aufbauen. Außerdem experimentieren wir zum einen mit Variationen, z.B. rekursive Regelanwendung, *lattice phrase extraction*, bei der wir unsere *phrase table* aus der umgeordneten Quellseite des Trainingscorpus aufbauen, zum anderen mit verschiedenen Ansätzen zum Parsen.

Wir testen unsere Systeme auf einem Satz von Wikipedia-Artikeln zum Thema Kyoto. Zur Evaluation unserer Experimente nutzen wir BLEU und RIBES für eine umfassende Einschätzung unserer Konfigurationen, so wie Kendall's τ und *chunk fragmentation*, eine Metrik, die die Anzahl der Satzfragmente misst, die nicht korrekt angeordnet sind, um eine Einschätzung der *reordering*-Qualität alleine zu erhalten. Weiterhin führen wir eine manuelle Analyse auf 100 Sätzen durch, um zu sehen, ob unsere Systeme in der Lage sind, das Verb ans Ende des Satzes zu schieben. Wir melden eine maximale Verbesserung von 1.95 BLEU-Punkten über ein System ohne Regeln. Wir zeigen außerdem, dass der Einsatz von *discontinuous reordering rules* einen starken Einfluss auf die essentielle Bedingung, das Verb an das Satzende zu setzen, hat. Darüber hinaus können wir berichten, dass konstituentenbasiertes Parsen dependenzenbasiertes Parsen als Basis der *reordering*-Regeln übertrifft und dass das es die Qualität verbessert, Regeln aus einem breiten Spektrum zu lernen. Mit einer manuellen Analyse zeigen wir, dass unsere Systeme *lattices* erzeugen, die in 85% der Fälle einen Satz mit dem Verb am Ende beinhalten.

Abstract

In recent years, machine translation, i.e. translation of text from one language to another by a computer, has become an important complement in our daily lives. Despite its flaws like incorrect or sometimes unnatural translations, machine translation (MT) is popular where human translators are too expensive. While the translation of words can be easily managed with a dictionary, reordering remains a major challenge in machine translation. Especially phrase based approaches are having trouble with this because they do not work with syntactic information on their own.
In MT, a recent approach to tackle this problem is to automatically learn reordering rules and apply them prior to decoding. Lately, this approach has been extended by a refined reordering model that uses information from syntax trees to determine possible reorderings. This model has successfully been tested on two European language pairs. In this thesis we apply this approach to a very distant language pair: English and Japanese. Japanese is a subject-object-verb (SOV) language meaning that the subject is at foremost position, followed by the object, which itself is succeeded by the verb. While OSV sentences are possible as well, it is mandatory that the verb is at the last position. This is an easy task for a rule-based MT system, but in statistical machine translation(SMT) we have to develop a model. We compare solely part of speech (POS) based rules with rules that also deploy syntax trees. Additionally, we experiment with variations such as recursive rule application, lattice phrase extraction, where we build our phrase table using the reordered source side of the training corpus, as well as different approaches to parsing.
We test our experiments on a set Wikipedia articles related to "Kyoto".

For the evaluation of our experiments, we use four automatic metrics: We use BLEU and RIBES to obtain a comprehensive estimation of our configurations. Moreover, we use Kendall's τ and chunk fragmentation, a metric that examines the number of chunks that are not ordered correctly, to get an estimation of the reordering quality alone. In addition to that, we perform a manual analysis of a set of 100 sentences to see whether our systems are able to place the verb at the end of the sentence. We report a maximum improvement of 1.95 BLEU points over a plain system. Besides, we show that the use of discontinuous reordering rules has a strong impact on the essential condition to shift the verb to the end of the sentence. We can further report that constituency parsing outperforms dependency parsing as a foundation for our reordering rules and that learning rules from a broad domain increases quality. With a manual analysis we show that our systems can provide lattices that offer a sentence with the verb at the end in 85% of the cases.

Contents

1. Introduction

Automatic translation of text has become an integral part of our daily lives within recent years. While human translators remain a crucial necessity in many professional areas where the production and translation of foreign text is mandatory, automatic translation gains relevance in areas where human translators are not considered because they are either too expensive or their service is deemed unnecessary. For scientific conferences, for instance, or when reading websites in a foreign language, the question is not whether to consult a human translator or use a machine translation system, but whether to have a cheap translation or none at all. When translating text with a statistical machine translation system, systems face the issue of word reordering. A good translation requires every word to be not only translated correctly, but also reordered to the right position. When translating to English, the expression of time is usually shifted to the beginning or end of the sentence. A rule-based machine translation system would use predefined hand made rules that recognise the expression of time via part of speech tagging or keywords and reorder the sentence. A statistical machine translation system on the other hand does not work with handmade rules. The only reordering in a simple statistical machine translation system is done in the decoder, but since this takes much time, various methods have been proposed to address this issue prior to decoding. A common method is to learn a reordering model and pass an already reordered source side to the decoder. This is sometimes referred to as 'preordering'. Recently, an approach has been proposed that provides a variety of possible reorderings to the decoder by building a lattice of possible reorderings from the original source sentence. This approach has been tested on European language pairs which are in terms of grammar relatively alike compared to more distant languages. Thus, the idea was to test the approach on a more distant pair of languages: English and Japanese. The Japanese language consists of three alphabets which allows to express the same sound with multiple characters. Even though there is a convention which character set to use for which word, within the large set of Kanji characters there are a few ambiguities. However, since the possibility to have two writings for one word is reasonably rare, we did not handle it. English and French, the language pair where the approach has been tested on, are both SVO(subject-verb-object) languages, i.e. the verb is between subject, which comes first, and object. Japanese, however, is an SOV language meaning that the verb is always at the end of the sentence. This fundamental difference in word order introduces an additional challenge on which we wanted to test the method.

1.1 Overview

In Chapter 2, we explain the fundamentals of Machine Translation. First, we will explain how a statistical machine translation system is built by explaining the intuition of statistical machine translation and introducing the various models, as well as fitting everything together in the decoder. Since this thesis deals with reordering, we will also discuss the most prominent approaches used in reordering. After that, we will exemplify the different evaluation metrics that we will use later on. We will pay special regard to the intuition behind the individual metrics to establish understanding why we use so many.
Chapter 3 will introduce various approaches that have been taken in the field. We will report important works that tackle the reordering problem, as well as papers dealing with the English-Japanese language pair. Also, we will describe the work that led to the systems we are using in the thesis on hand.
In Chapter 4, we describe how our systems modelled reordering. We will start by explaining the data we used for training and testing our models. Then, we will describe the respective experiments.
The results of the experiments are presented and discussed in Chapter 5.
In Chapter 6, we summarise our findings and give suggestions on additional experiments to improve translation quality even further.

2. Machine Translation

2.1 General Description

Machine translation seeks to automatically translate a source sentence F into a target sentence E. Statistical machine translation (SMT) tries to achieve this by using only statistical information: In training, probabilistic models are generated. Then, while translating, many possible translations, so called hypotheses or candidates, are generated on the basis of these models and the most likely hypothesis, subject to these models, is chosen.

$$P(e|f) = \frac{P(f|e) * P(e)}{P(f)} \tag{2.1}$$

The foundation of statistical machine translation is Bayes' theorem, as shown in Equation 2.1. It allows us to express the probability that sentence f translates to sentence e as a combination of other probabilities: The probability of sentence e or f alone and the probability that sentence f is a translation of sentence e. The goal is to find the sentence $\tilde{e}$ that maximizes the formula. Since the probability of the given sentence $P(f)$ in the denominator does not depend on any candidate e, we come to the formula

$$\arg\max_e P(f|e) * P(e).$$

$P(e)$ will be realized by the language model, $P(f|e)$ by the translation model.

2.2 Language Model

A language model is used to assign a probability to any sequence of words. The foundation of any language model is a preferably large sample of that language. This sample is usually given in the form of sentences that have been used in some context[1] and revolve around a certain topic. While common words will appear with

[1]e.g. newspaper articles, conference material, wikipedia entries

...	...
あなやま	0.01677
あに	0.01677
あね	0.02298
...	..

Figure 2.1: Schema of a language model.

similar frequency in different text corpora, the frequency of topic specific terminology is highly dependent on the topic of the corpus. It is therefore important to choose the right corpus for training the language model to model the jargon of the translation task. To adapt existing language models to the topic of the task, it is an easy measure to interpolate them. If we compute the likelihood of a n-word sequence, i.e. an n-gram solely as the fraction $\frac{\#\text{occurrances of n-gram in corpus}}{\#\text{n-grams in corpus}}$, sequences not contained in the corpus will obtain a likelihood of zero. This can be the case because a word in the sequence has not been encountered in training or because this specific order of words has not been observed. To improve this situation, smoothing is used: to avoid assigning a zero probability, every observed n-gram donates a certain amount of probability mass to use for n-grams we have not seen yet. Figure 2.1 shows a possible form of a language model: observed n-grams are on the right side and their respective probability on the left. In practice, language models do not store the probability, but its negative binary logarithm in order to avoid near zero values and exchange multiplication for the computationally safer addition. Also additional weights for smoothing might be stored.

2.3 Alignment Model and Translation Model

...	...	...
かきます	to write	0.8
かけます	to hang	0.3
かけます	to lock	0.6
...	...	...

Figure 2.2: Schema of a translation model

The translation model is the heart of every translation system. It provides a word to word mapping from source to target language along with probabilities assigned to each translation. Figure 2.2 shows an example of a translation model. Such a model alone can already provide a glossing, i.e. an automatic word for word translation which for close language pairs can already give an idea what the text is about. Since words usually change position during translation, an alignment model is introduced to provide a probability distribution for each word position in the source sentence to any position in the target sentence. Since manually aligned parallel corpora are rare (there is no use case where alignment would be a by-product), the alignment has to be estimated. The estimation, however, is a chicken and egg problem: In order to know which target word a source word aligns to, we need a lexicon that provides us with possible translations. This lexicon model, though, can only be built if we have an alignment that provides us with target words for a source word. This problem is solved with the expectation-maximization algorithm where we initially assume an alignment that maps source word i to target word j. Now a lexicon is created on that alignment and, with the help of this lexicon, a more accurate alignment is estimated. This is repeated until convergence.

2.4 Phrase Translation Table

Phrase based translation systems use translation tables that map whole sequences of words, i.e. phrases. The idea is to translate whole chunks of text to a sequence of words that we know is natural (because we extracted it from real world examples) and provide more context to choose the right translation. If we look again at Figure 2.2, we see that *'かけます'* can mean *'to hang'* as well as *'to lock'*.

While we see from the word based translation table that *"to lock"* is overall more likely in the corpora used for training, the 2 words that precede *'かけます'*, ideally the object *'かけます'* refers to, could give us valuable information how to translate the phrase.

2.4.1 Phrase Extraction

Previously, we referred to phrases as a sequence of words. It is important to note that phrases in the context of SMT are not restricted to linguistic phrases since our main goal is to get more contextual information and not to segment the sentence linguistically. We cannot extract every subsequence of a sentence, but only those for which we can determine an alignment to a phrase of the target sentence. We define a pair of subsequences from source and target (E_n, F_m) with $E_n = e_1, ..., e_n, F_m = f_1, ..., f_m)$ as consistent with an alignment A if there is at least one alignment between the sets and no word is aligned to a word outside the other set. Mathematically, we can write this as:

$$\begin{aligned}
&\exists e_i \in E, f_j \in F : (e_i, f_j) \in A \\
&\forall e_i \in E, (e_i, f_j) \in A : f_j \in F \\
&\forall f_j \in F, (e_i, f_j) \in A : e_i \in E
\end{aligned}$$

F and E are the source and target sentences, A is the alignment between them and f_j and e_i are the respective words.

Figure 2.3 shows an example phrase extraction from a sentence. While *"My dog also"* is an intuitive choice, we decided to highlight *"also likes to eat sausage"*. Although not contiguous, it is a consistent alignment: No word of either source or target phrase is aligned to a word outside the alignment box and the box is non-empty. The whole sentence or any word pair also qualify as a consistent alignment.

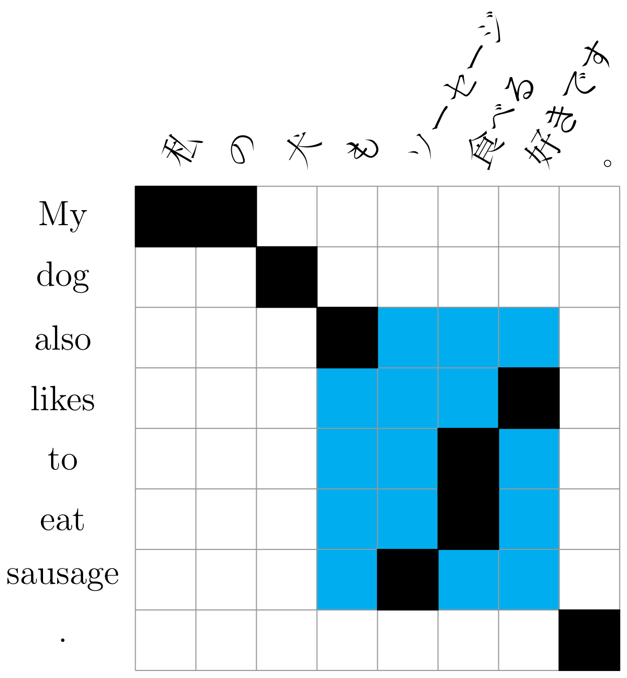

Figure 2.3: An instance of an alignment matrix, with one phrase pair highlighted

2.4.2 Phrase Scoring

A simple way of assigning probabilities to phrases is to use maximum likelihood, as we did in the translation model:

$$\phi(\bar{f}|\bar{e}) = \frac{\text{count}(\bar{e}, \bar{f})}{\sum\limits_{f_i} \text{count}(\bar{e}, \bar{f}_i)},$$

where $\text{count}(a, b)$ is the number of sentences in the training corpus where phrase a is aligned to phrase b. The phrase pair $\bar{e}$, $\bar{f}$ gets a conditional probability as the fraction of the count of phrase pair $\bar{e}, \bar{f}$ by the count of any pair with $\bar{e}$. However, this measure is very imprecise as it pays no attention to the length of the phrases for instance. To refine the scoring of phrases a variety of models is used, four of which we introduce in the following.

2.4.2.1 Lexical Weights

If a phrase pair is encountered only once, its conditional probability is 1. The high probability in this case is misleading, as it suggests a high certainty, but is only due to a lack of occurrences in the training data. To smooth such cases down, we introduce a lexical weight model that falls back on the word based translation model. Repetition of words is much more frequent than repetition of phrases, so the certainty here is much higher.

$$lex(\bar{e}|\bar{f}, a) = \prod_{i=1}^{I} \frac{\sum\limits_{\forall(i,j) \in a} w(e_i, f_j)}{|(i, j) \in a|} \tag{2.2}$$

Equation 2.2 shows a formula for lexical weighting. Given the phrase pair $\bar{e}, \bar{f}$ and the alignment a between them, we multiply the averaged translation probabilities for every word in the target phrase $\bar{e}$. As average translation probability, we define the sum of translation probabilities $p(e_i|f_j)$ from the word based translation model, over all f_j in the source phrase that align to e_i.

2.4.2.2 Distance-Based Reordering Model

A reordering model is introduced to penalise misplacement of phrases on the target side. The models' intuition is to avoid jumps in the alignment of phrases on the target side. Therefore, it favours a phrase by phrase translation and even penalises reordering rules i.e. shifting the verb to the end, which we need for a natural translation. Such rules will be addressed in Section 2.7.

$$p(\bar{f}|\bar{e}) = \prod_{i=1}^{I} d(\text{start}_i - \text{end}_{i-1} - 1) \tag{2.3}$$

Equation 2.3 shows the formula for scoring the reordering of a sentence e. For every phrase f_i from the source sentence F, we look at start_i, the position of the first word in e_i, and end_{i-1}, the position of the last word in e_{i-1}, where e_i is always the target phrase aligned to f_i.

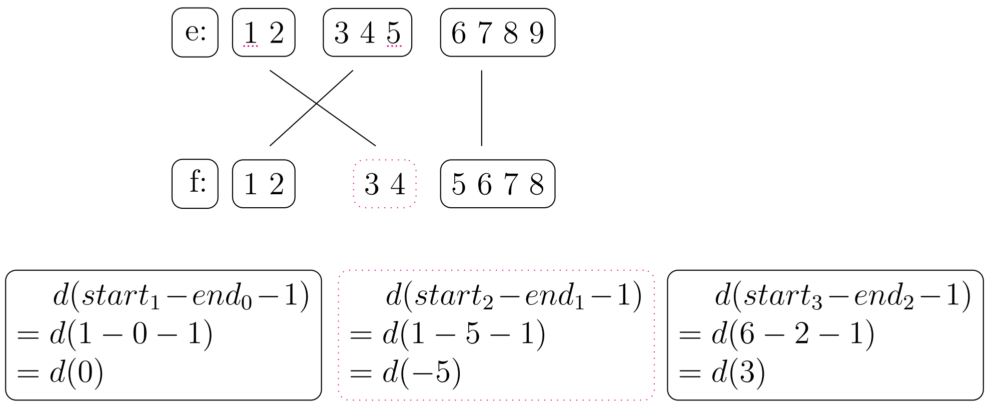
e:
1 2
3 4 5
6 7 8 9
f:
1 2
3 4
5 6 7 8
$d(start_1 - end_0 - 1) = d(1 - 0 - 1) = d(0)$
$d(start_2 - end_1 - 1) = d(1 - 5 - 1) = d(-5)$
$d(start_3 - end_2 - 1) = d(6 - 2 - 1) = d(3)$

Figure 2.4: A simple example for a distance-based reordering model

Figure 2.4 illustrates how the score is obtained. The words have been exchanged for their position names for convenience. For the source phrase f_2 *"3 4"*, we look at end_0 and $start_1$. The preceding phrase of f_2, f_1 is aligned to e_2, so the position of the last word in e_2,*5*, is end_0. Next, we look at $start_2$, the first word of the target phrase aligned to f_2. f_2 translates to e_1, so $start_2$ is the first position in e_1,*1*. Thus, we obtain a score of -5 for the second phrase. Phrase 1 gets a score of 0 since there is no reordering in relation to the 0th phrase. At last, we need to define $d()$. While we could define d as a probability function dependent on the involved phrases, an exponentially falling function will suffice. We define $d(x) := a^{|x|}$ with $0 < a < 1$. So if we choose $a = 0.8$, we get a reordering score of $d(0) * d(-5) * d(3) = 0.8^{|0|+|-5|+|3|} \approx 0.1678$.

2.4.2.3 Lexicalized Reordering Model

While the distance-based reordering model scores reordering regardless of the phrases themselves, based on shift distance alone, we still need another model to score reordering based on phrases. Thus, we introduce the lexicalized reordering model. With this model, we estimate the probability for any phrase to be reordered based on its content alone. We define 3 reordering orientations: *monotone*, *swap*, and *discontinuous*.

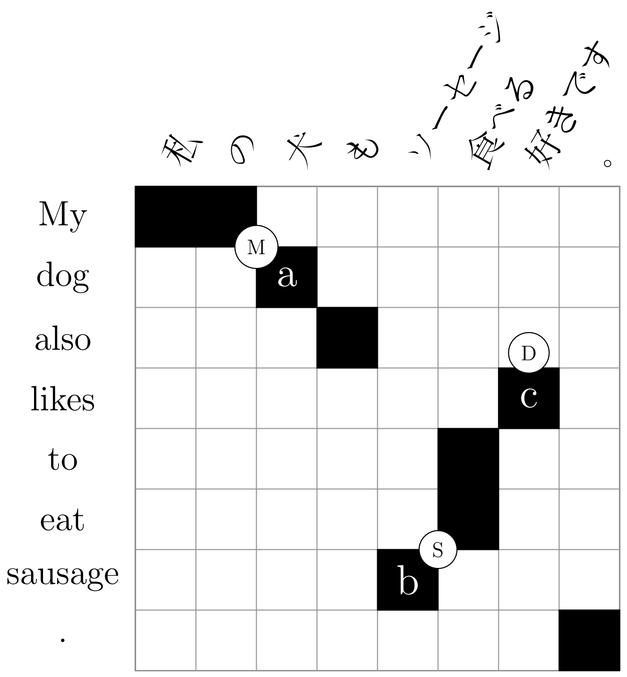

Figure 2.5: An example for the 3 types of orientation: M*onotone*, D*iscontinuous* and S*wap*.

Figure 2.5 shows these three types. We say a phrase pair is reordered *monotone* when it has an alignment point at the top left, such as phrase pair *a*. A *swap* occurs when there is an alignment point at the top right, as with phrase pair *b*; and if there is no alignment point at the top, we have a *discontinuous* orientation. The count of occurrence is determined during phrase extraction. Probabilities are calculated via maximum likelihood, similar to the translation probabilities of the phrases themselves.

$$p(\text{orientation}|\bar{f},\bar{e}) = \frac{count(\bar{f},\bar{e},\text{orientation})}{count(\bar{f},\bar{e})}$$

The problem of data sparsity can be reduced by smoothing with the overall probability of the orientation. Koehn (2010) present 2.4 as an example,

$$p(\text{orientation}|\bar{f},\bar{e}) = \frac{\delta p(\text{orientation}) + count(\bar{f},\bar{e},\text{orientation})}{\delta + count(\bar{f},\bar{e})} \tag{2.4}$$

p(orientation) being the overall probability of orientation over all phrases.

2.4.2.4 Word Penalty

To penalise hypotheses with too many or too few words, a factor ω is introduced that is multiplied with itself for every word in the hypothesis, yielding $\omega^{|\text{words}|}$ as score. The right value is determined by tuning. An $\omega < 1$ penalises long sentences, $\omega > 1$ penalises short sentences.

2.4.2.5 Phrase Penalty

A phrase penalty is used to score the number of phrases a sentence is segmented into. Similar to the word penalty, a factor p is taken to the power of the number of phrases in the sentence: $p^{|\text{phrases}|}$. $p < 1$ penalises many, i.e. short phrases, $p > 1$ penalises few, i.e. long phrases.

2.5 Log-Linear Model

In section 2.1, we defined the search problem as

$$\arg\max_e P(f|e) * P(e),$$

where we realised $P(f|e)$ as our translation model (TM) and $P(e)$ as our language model (LM). But in order to create these models efficiently, we made some simplifications. Our n-gram language model for instance, takes only preceding words into account when computing a probability. In order to even that out, weights for language model and translation model are introduced which leads us to

$$\arg\max_e P(f|e)^{\lambda_{LM}} * P(e)^{\lambda_{TM}}.$$

To incorporate the other models, we abandon the Bayes' theorem entirely and substitute a log linear model. First, we multiply other models with weights, resulting in

$$\arg\max_e \prod_i m_i^{\lambda_i}$$

where m_i is the output of model i and λ_i the respective feature weight. To avoid underflows, the formula is further rewritten to

$$\arg\max_e exp(\sum_i log(m_i) * \lambda_i) = \arg\max_e \sum_i log(m_i) * \lambda_i$$

This formula will be the foundation for evaluating any hypothesis in the decoding step.

2.6 Decoder

Essentially, the Decoder uses the different models to create different hypotheses for the source sentence, score them and choose the best one. We will shortly explain

how this works.
In a very simple translation system with a word-based translation model, the decoder simply translates the source sentence word by word. This results in one hypothesis being chosen as translation. However, a good translation model provides translation of whole phrase pairs (n-grams) and multiple possible translations for the same phrase. This leads to two sources for multiple hypotheses: on the one hand, there are several ways to segment a sentence into phrases, on the other hand, multiple translations exist for the same phrase. In order to choose the best hypothesis, each hypothesis is attributed a score. Previously created models are used for determining this score. The translation model, for example, provides a score for the translated phrase and the language model evaluates how 'natural' and fluent a hypothesis is. Additionally, a variety of other so called features can be evaluated, e.g. the ratio of the number of words in the hypothesis and in the source sentence.

2.6.1 Minimum Error Rate Training

Minimum error rate training (MERT) in SMT has been introduced by Och (2003). The basic idea is that the decoder generates a list of N candidate translations and chooses the best one by analysing each candidate translation for a set of features, summing up those feature values, and selecting the one with the highest score. By feature we mean a number, typically between 0 and 1, that is the output of a model. Some features can be more important than others, depending on the specific translation task and domain, thus a weighted sum is taken with feature values and weights ranging from 0 to 1 for convenience. As the importance of each feature varies with the task, the goal is to calculate proper feature weights. To do so, an optimization step is performed in which a translation is carried out on a development set that is similar to the test set on which the actual translation will be performed. Ideally, development and test set are subsets of the same corpus. Since we know the reference translations for each sentence from the development set, k-dimensional optimization is performed on the n-best lists to find the k weights that lead to the selection of the most promising translation candidates, i.e. the set of translations that get the highest score from the specified evaluation metric.

2.7 Reordering

The methods introduced so far enable us to build a translation system that performs reasonably well on European languages. However, for distant language pairs such as English-Japanese, we need means to effectively reorder the source sentence. While the decoder allows reordering to a limited degree, sophisticated rules such as shifting a certain words to the end or switching not-consecutive phrases cannot be realized. It would be possible to set the reordering window as infinite, but then the number of hypotheses would explode. In this section, we look at different approaches to reorder a sentence during translation. We will first regard reordering within phrases, which is realized through the phrase table. Then, we will deal with lexicalized reordering which we have already seen in Section 2.4.2.3. Afterwards, we will introduce the concept of reordering the source side prior to decoding, which is followed by reordering during decoding time. This section is concluded by a glance at parsing sentences as an addition to purely statistical methods.

2.7.1 Reordering within Phrases

Reordering within phrases is handled implicitly through the phrase table. If we take the English term *"European Union"* and its French counterpart *"L'Union européenne"*, we see that the tokens for *"Europe"* and *"union"* simply switch places. If this term appears in the test set, we can expect it to be captured in training, given that test and training data have the same topic. However, this approach is unreliable towards phrases with words that do not necessarily appear together. For instance, the term *"an interesting book"* features two words, *"interesting"* and *"book"*, that can appear in many word pairs. There is a good chance that we observe many phrases with *"interesting"* objects and possibly many descriptions of a *"book"*, though not in this particular combination. In that case, we have to resort to higher level approaches.

2.7.2 Lexicalized Reordering

In Section 2.4.2.3, we introduced a model that provides probabilities given a phrase pair within a sentence and an orientation type. During decoding, this model can be used to score hypotheses in order to discard unlikely ones.

2.7.3 Reordering of the Source then Monotone Translation

A popular approach is to reorder the source side prior to decoding. The advantage here is that the reordering in the decoder addressed in Section 2.7.4 can be restricted to a smaller window or even abandoned at all, allowing for faster decoding. This approach is the main focus in this thesis.

2.7.4 Reordering During Decoding with Jumps

The decoder allows reordering to a limited degree. Instead of monotone translation we can use a sliding window to allow the decoder to process any word within the window instead of the immediate next word.

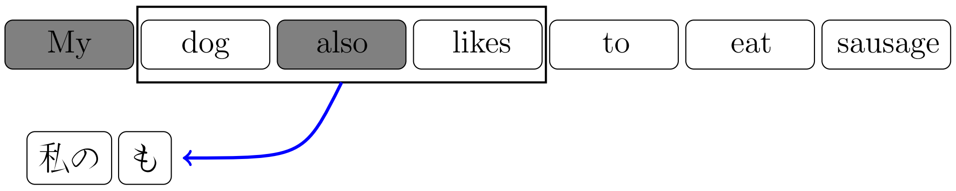

Figure 2.6: An example of a sliding window of size 3 in the decoding process

Figure 2.6 shows an example: The first word has already been translated, hence, the sliding window has advanced to the second word. Now any word or n-gram in the window, here *'also'*, can be translated and directly appended to the hypothesis. Therefore *'dog'* is skipped but it has to be translated eventually since the window cannot move on any sooner.

2.7.5 Syntax Approaches to Generate Grammatical Target Sentence

Another way to generate a grammatical target sentences is to augment the decoder with a syntactic parser. In string-to-tree systems, for instance, the target sentences of the training data are parsed into tree structures. Subsequently, pairs of subtrees and phrases are extracted. During decoding, such subtrees are connected to one target tree. Here, the idea is that, given an reliable parser, the resulting target trees are automatically grammatical.

2.8 Evaluation

Evaluation usually compares a translation by a machine translation system with a human generated reference translation. However, for better precision we will also compare alignments. An automatic evaluation by a computer is usually preferable over human evaluation. It can give fast, reproducible, and objective results. The performance does not decrease over time and it is inexpensive compared to a human evaluator. The advantage of a human evaluator, who actually understands the meaning of the sentence and the exact type of error the system makes, is easily outweighed by those factors.

In the following, we present the metrics needed for automatic evaluation in this thesis. The first two metrics evaluate translation quality, while the subsequent two measure reordering quality. After that, we will shortly discuss manual evaluation, because we performed it on a reasonable subset of sentences to gain more insight into the reordering process.

2.8.1 BLEU

BLEU (*"Bilingual Evaluation Understudy"*) was introduced by Papineni et al. in 2002 as an automatic evaluation metric that should mimic human evaluators but, at the same time, be quick and affordable. It is the most commonly used automatic evaluation metric in SMT. The idea is to measure the frequency with which n-grams (i.e. a sequence of n words) in the candidate translation appear in a reference translation for that candidate. This frequency is measured with a modified form of precision measure. It is possible to work with a set of possible reference translations per candidate to allow for multiple correct translations of the same word.

2.8.1.1 Precision

Precision in the context of classification is defined as the number of true positives by the number of positives. So if we have a given set A and a system tries to classify the members $a*$ of a subset $A*$ of A that fulfill a certain condition (i.e. are true) the precision is the number of items which are correctly labelled as $a*$ (the true positives) divided by the number of items of A that are labelled as $a*$ (the positives).

Figure 2.7 shows an example precision for a classifier that is meant to recognise triangles.

In the context of machine translation we want to compare words in the candidate with words in the reference, which is essentially precision: The true positives are

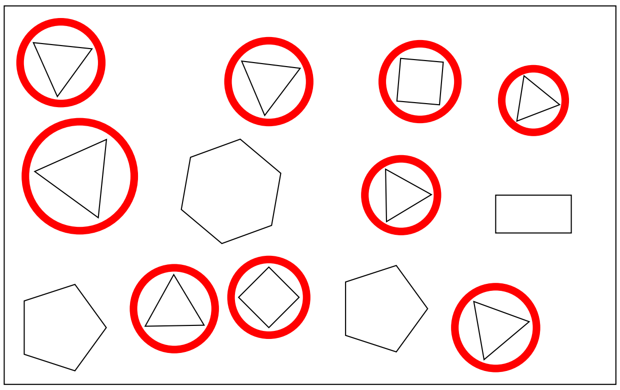

Figure 2.7: A classifier recognises 7 triangles and 2 other polygons as triangles. It has therefore a precision of $\frac{7}{9}$

words which are in the candidate and can be found in any reference translation while the positives are the words in the candidate. Hence, we get the formula

$$PRECISION_{adapted} = \frac{\sum\limits_{c \in C} count(c, C) * \delta(c \in R)}{\sum\limits_{c \in C} count(c, C)} \tag{2.5}$$

The numerator denotes the number of words from the candidate that can be found in any reference. The denominator denotes the number of words in the candidate. δ serves as a decision function that translates the boolean values *true*, i.e. $c \in R$ and *false*, i.e. $c \notin R$ into 1 and 0, respectively.

2.8.1.2 Modified Precision

Precision is intended to evaluate a classification task, i.e. to evaluate how many given items were correctly recognised. In Machine Translation however, we have a generative task. A SMT system generates candidates and we have to label them into correct ones and incorrect ones afterwards. So if the candidate contains the word ‘a’ twice but the reference contains it only once, we do not know which one is correct. In the above attempt of transferring precision to SMT it has been decided that every word in the candidate is correct if it can be found in any reference. This can yield good scores for actually poor candidates as in Figure 2.8 where the word ‘a’ has clearly been overgenerated.

cand:	an apple a day a a a
ref1:	an apple a day keeps the doctor away
ref2:	an apple a day keeps doctors away

Figure 2.8: The candidate translation has overgenerated the phrase ‘a’.

In the above example, the word ‘a’ appears four times while it is found not more than once in any of the reference translations. However, it contributes four times positively to the score. This leads to an unjustifiably high score of 1. One would expect the 2nd to 4th occurrences to be treated as false positives since there is no 2nd occurrence of it in any reference translation.

So the intuitive solution to this is that a word w is not allowed to score more often than it appears in any reference. Therefore its occurrence is clipped down by the maximum times it appears in any reference translation. We define clipped as

$$count_{clipped}(w, C) = Min(Count_C(w), \max_{r \in R}(Count_r(w)))$$

and a modified precision as

$$precision_{mod}(C) = \frac{\sum_{c \in C} count_{clipped}(c, C) * \delta(c \in R)}{\sum_{c \in C} count(c, C)} = \frac{\sum_{c \in C} count_{clipped}(c, C)}{\sum_{c \in C} count(c, C)}$$

The decision function δ now becomes obsolete as the check for occurrence is handled by $count_{clipped}$. Thus, the sentence from Figure 2.8 gets a modified precision score of $\frac{4}{7}$.

In order to compute the precision of a whole corpus of candidates the clipped count of all words per candidate is summed up over all candidates and divided by the overall number of words in the corpus. This gives us the following formula[2]

$$precision_{mod}(corpus) = \frac{\sum_{C \in corpus} \sum_{c \in C} count_{clipped}(c, C)}{\sum_{C \in corpus} \sum_{c \in C} count(c, C)} \tag{2.6}$$

2.8.1.3 n-gram Precision

So far we can evaluate the use of right words, i.e. adequacy, but not their order, i.e. fluency. If we sort the example sentence lexically, we obtain the same modified precision score. To penalise such a lack of fluency we also look at sequences of n words. Thus, the formula for precision is generalized to:

$$precision_{n\text{-}gram}(corpus) = \frac{\sum_{C \in corpus} \sum_{n\text{-}gram \in C} count_{clipped}(n\text{-}gram, C)}{\sum_{C \in corpus} \sum_{n\text{-}gram \in C} count(n\text{-}gram, C)} \tag{2.7}$$

	$precision$	$precision_{mod}$	$precision_{n\text{-}gram}$
a an apple away day doctor keeps the	1	1	$\frac{10}{21}$
an apple a day keeps the doctor away	1	1	1

Table 2.1: n-gram precision for a candidate that has the wrong order, with a maximum of 3-grams.

Table 2.1 depicts the modified n-gram precisions for a senseless permutation of the reference from Figure 2.8. In the second line, the reference itself is also scored for

[2]One might be tempted to average over the modified precision for each sentence, but this would weight each sentence equally, regardless of their length. Therefore, the average over all words is taken.

comparison. The columns $precision$, $precision_{mod}$, $precision_{n\text{-}gram}$ correspond to the Equations 2.5, 2.6, 2.7. We use up to 3-grams in this example. While every unigram can be found in the reference, only two bigrams and zero trigrams can be found. If we were to account for 4- and 5-grams as well, the score would further decrease, since zero matching 3-grams implies zero matches for all higher n-grams. The values for n are commonly 4 or 5 at most since higher values do not seem to gain enough information to justify the additional computing time.

2.8.1.4 Brevity Penalty

It is further desirable that the candidate translation matches its reference translations in length. Too many words in the candidate are already penalised through the use of precision measure, but short candidates can still obtain unjustifiably high scores. A candidate that consists of an arbitrarily small sequence of a reference translation would even get a precision of 1 since every n-gram can be found in a reference. Therefore, an external penalizing factor is used. To allow for some freedom on sentence level the brevity penalty is computed on the whole corpus and not for each sentence individually. If the candidate is not shorter than the reference, it should equal 1 in order to have no effect. If it is shorter, it should decrease. Papineni et al. (2002) decided on an exponential decrease relative to $\frac{r}{c}$ where c is the sum of all words in all candidates and r is the sum of the references that match the corresponding candidate best in length. They get the following formula:

$$BP = \begin{cases} 1 & c > r \\ e^{1-\frac{r}{c}} & c \leq r \end{cases}$$

2.8.1.5 Combination of n-gram Precisions

Since the various n-gram precisions have very different ranges (a high 5-gram precision is rather unlikely and a k+1-gram always implies 2 k-grams) the geometric mean($\sqrt[n]{p_1 * p_2 * ... * p_n}$) is more appropriate than the arithmetic mean. Thus, the formula is $BLEU = BP * \sqrt[n]{p_1 * p_2 * .. * p_5}$.

2.8.2 RIBES

Since BLEU accounts for global word order only by n-grams, the RIBES (*"Rank-based Intuitive Bilingual Evaluation Score"*) (Isozaki, Hirao, et al., 2010) metric has been established for distant language pairs such as English and Japanese in an effort to explicitly address word order.

$$RIBES = \sum_{c \in C} \frac{nkt * precision^{\alpha} * BP^{\beta}}{\#\text{Words in c}}$$

As in BLEU, the Brevity penalty is $BP = \min\{1, e^{1-\frac{r}{c}}\}$, r being the number of words in the reference and c being the number of words in the hypothesis. Kendall's τ was introduced by Kendall (1938) as a mean to compare orderings by different subjects. It computes as

$$\frac{\#\text{increasing pairs} - \#\text{decreasing pairs}}{\#\text{all pairs}}.$$

We assume a reference order of $1, 2, ..., n$ and a permutation of this sequence generated by the subject. Now it is determined for every pair of ranks in the permutation whether the ranks are increasing as in the reference and are, thus, correctly ordered, or whether they are decreasing. A more in-depth analysis of Kendall's τ will be given in 2.8.3. The introduced version of Kendall's τ ranges between -1 and 1. In order to keep the score positive, Isozaki, Hirao, et al. use normalized Kendall's τ which computes as

$$nkt = \frac{\#\text{increasing pairs}}{\#\text{pairs}}.$$

Precision is the number of words from the hypothesis that can be found in the reference divided by the number of words in the hypothesis. α and β are parameters with values between 0 and 1, in our case 0.25 and 0.1 were used.

2.8.3 Kendall's τ

BLEU and RIBES compare the output of the translation system with a reference in the source language. Hence, many factors influence the score, for instance alternate translations. The hypothesis *"This is my home"* for reference *"This is my house"* will have a lower score because the metric can only score *"home"* as a translation error although the reordering is correct. The next two metrics have been used to solely examine the quality of reordering. We used Kendall's τ as described in Neubig, Watanabe, & Mori (2012). Kandall's τ is a measure for the accuracy of pairwise ordering, i.e. the ratio of the sum of correctly ordered pairs and the sum of pairs which can be written as

$$\tau = 1 - \frac{\#\text{incorrectly ordered pairs}}{\#\text{all possible pairs}}.$$

A few changes(ties, unaligned) have to be made to apply this metric to translations. For every word f'_j a_j is the set of indices in the target sentence that f'_j is aligned to. For a_j we define a_{j_1} as the first index in a_j and $a_{j_\$}$ as the last index. With these sets, a ranking function $r(f_j)$ can be defined. For two words f_j and f'_{j+1}, $r(f'_j) < r(f'_{j+1})$ holds if $a_{j_1} < a_{j+1_1} \wedge a_{j_\$} \leq a_{j+1_\$}$ or $a_{j_1} \leq a_{j+1_1} \wedge a_{j_\$} < a_{j+1_\$}$.
$r(f'_j) = r(f'_{j+1})$ holds if neither $r(f'_j) < r(f'_{j+1})$ nor $r(f'_{j+1}) < r(f'_j)$ hold. Since the above formula for Kandall's τ was made assuming a strict order, we subtract the number of words, which have the same rank assigned, from the denominator and come to an adjusted formula:

$$\tau = 1 - \frac{\sum_{i=1}^{J-1} \sum_{j=i+1}^{J} \delta(r(f'_i) > r(f'_j))}{\sum_{i<j \in J} \delta(r(f'_i) \neq r(f'_j))}.$$

Words with no alignment are assigned the same rank as the next word to the right or left, depending on the type of language. Neubig, Watanabe, & Mori (2012) suggest to group such words to the next word on the left for head-final languages, such as Japanese, and to the right for head-initial languages, such as English. We follow this approach.

An example score can be seen in Figure 2.9.

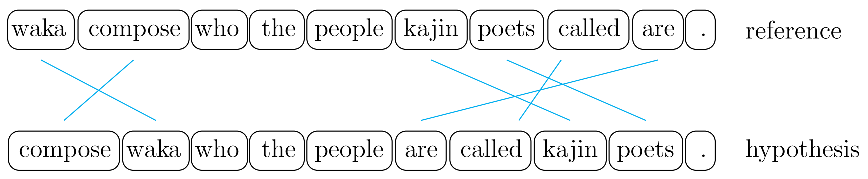

Figure 2.9: There are a total of six pairs where the respective words in the reference have a different order. With a total of 45 pairs we get an accuracy of $1 - \frac{6}{45} = \frac{13}{15}$

2.8.4 Chunk Fragmentation

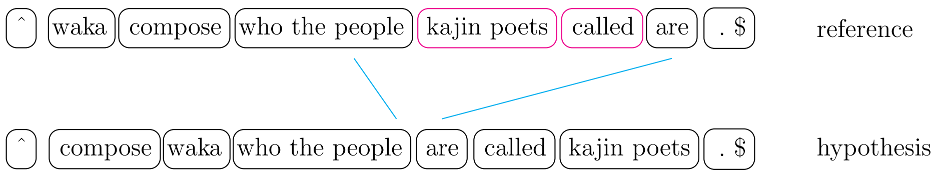

Figure 2.10: An example for chunk fragmentation. There is a chunk border between 'people' and 'are' because their corresponding ranks in the reference differ by more than one.

While chunk fragmentation has been described in earlier works, we use the version described in Neubig, Watanabe, & Mori (2012), as it is explained very detailed there. Chunk fragmentation counts the number of times the reader has to jump in a sentence of J words in order to read it in the right order.
A jump occurs whenever the immediately following word f'_{j+1} has not the same or immediately following rank as f'_j. So an indicator function DISCONT is defined that returns 1 in the case of a jump:

$$DISCONT(f'_j, f'_{j+1}) = \begin{cases} 0 & r(f'_j) = r(f'_{j+1}) \lor r(f'_{j+1}) = r(f'_j) + 1 \\ 1 & else \end{cases}$$

We use the ranking function r that we defined in Section 2.8.3. It can easily be seen that two following jump points always delimit a chunk: If there are two or more chunks in between, there has to be at least one jumping point in between, and if the chunk exceeds the second jumping point, then there is no second jumping point. With the DISCONT function, we can just sum up all jumping points via $\sum_{j=1}^{J-1} DISCONT(f'_j, f'_{j+1})$. Since the first and last word of the reordered sentence are only compared once and are, hence, underrepresented, artificial sentence delimiters $f_\wedge$ and $f_\$$ are introduced with $r(f_\wedge) = 0$ and $r(f_\$) = \max_{F'} r(f'_j) + 1$. Thus, we obtain the accuracy formula:

$$1 - \frac{\sum_{j=0}^{J} DISCONT(f'_j, f'_{j+1})}{J+1},$$

where the denominator describes the number of comparisons.

Figure 2.10 shows an example sentence with chunks drawn in. With the helper tokens $f_\wedge$, $f_\$$ there are eight chunks, so the reader has to jump seven times. The sentence consists of ten words(twelve if we count delimiters), consequently, there are eleven comparisons which results in an accuracy of $1 - \frac{7}{11} = \frac{4}{11}$.

2.8.5 Manual Evaluation

While the automatic measures introduced so far can evaluate an enormous amount of hypotheses, there are also reasons to perform a manual evaluation. First of all, automatic evaluation metrics evaluate many features at once. A manual evaluation can be helpful to look at one particular feature alone, especially, if it is tedious to write it in code. Secondly, a human has an intuitive sense for the qualities that we expect from a good translation, such as adequacy and fluency, while a machine has no sense for that and has to fall back on statistical measures. Moreover, the eventual goal is to produce translation results that satisfy human requirements, so it is intuitive to use human evaluators additionally. Another advantage of human translators is their ability to generalize. While it is difficult enough to acquire parallel corpora, there is virtually no dataset with multiple reference translations, so an automatic evaluation has to assume that there is only one correct translation for every sentence. If we take the reference sentence *"I go home."* and a hypothesis *"I go to my house."*, a human evaluator could see that despite a little odd choice of words, the hypothesis is adequate, while an automatic metric could only determine that the hypothesis is comparatively long and one third of the reference is missing.

3. Related Work

In this Chapter, we will introduce different approaches that have been taken to address either reordering or the English-Japanese language pair. We will start with an overview of approaches regarding translation from and to Japanese. This is followed by important approaches in the topic of reordering. Finally, we will give an overview of the work that our systems are built upon.

Because Japanese and English are barely related as opposed to English and other European languages, rule-based systems are traditionally preferred for machine translation in this area. Nagao (1981) suggests working with analogies, i.e. mimicking the human process of using phrases known from different contexts. In addition to phrase extraction, an artificial thesaurus is built to deal with similarities. Sumita et al. (1990) compare this example-based approach with rule-based machine translation and point out that an example-based system is more robust and easier to adapt and maintain. For dealing with segmentation and morphology, Neubig, Watanabe, Mori, & Kawahara (2012) suggest to use an inverse transduction grammar framework and look at sentences on a character instead of a word level.

There are various approaches to capture and make use of the structure of sentences. A very interesting approach was developed by Chiang (2005). He uses synchronous context free grammars to capture the hierarchy of a sentence. Synchronous context free grammars extend context free grammars insofar that every rule has a source and target language. This is realized by extracting phrase pairs with wildcards. Since every wildcard can be replaced with a different phrase pair, the phrase pairs model a context free grammar. To reduce the number of extracted rules, a series of filters is applied, among them a filter which restricts rules to have only two nonterminal symbols. Parsing is done by a CKY parser in combination with beam search. An annotated source side is not needed, all rules are obtained based on frequencies.

Figure 3.1 shows a possible set of rules extracted from the sentence “Australia is one of the few states that have diplomatic relationships with North Korea.” and its Japanese translation. Note that the first two rules are not extracted but hard coded since they are needed to enable parsing in the first place.

Another approach are tree driven methods where the decoder works directly on parse trees. An early work in this area is Liu et al. (2006) who employ a syntactic parser

$S \rightarrow$	$\langle S_1 X_1,$	$S_1 X_1 \rangle$
$S \rightarrow$	$\langle X_1,$	$X_1 \rangle$
$X \rightarrow$	$\langle$is $X_1,$	X_1 ある$\rangle$
$X \rightarrow$	$\langle$have X_1 with $X_2,$	X_2 との X_1 持っている$\rangle$
$X \rightarrow$	$\langle X_1\ X_2,$	the X_1 that $X_2\rangle$
$X \rightarrow$	$\langle$one of $X_1,$	X_1 の一つで$\rangle$
$X \rightarrow$	$\langle$Australia,	オーストラリアは$\rangle$
$X \rightarrow$	$\langle$few countries,	数少ない国$\rangle$
$X \rightarrow$	$\langle$diplomatic relations,	外交関係$\rangle$
$X \rightarrow$	$\langle$North Korea,	北朝鮮$\rangle$

Figure 3.1: A possible set of rules from a synchronous context free grammar for English and Japanese.

on the source side and extract alignment information between tree nodes and target sentences. By mapping not only leaf nodes, i.e. words from the parse tree, but also non-leaf nodes, i.e. POS-tags, both, word and phrase level, are addressed. A dependency-to-string approach is introduced by Quirk et al. (2005).

A different approach in preordering is done by Isozaki, Sudoh, et al. (2010). We mentioned before that Japanese is a SOV language. Although its word order is generally very free, it is mandatory for the verb to be placed at the end of the sentence. Additional to the order of subject, object and verb, languages can also be categorized by their 'heads.' The head of a linguistic phrase is the word whose POS tag describes the whole phrase. So for a noun-phrase, the head is one of the contained nouns, and for every verb-phrase the head is a verb. Depending on the position of this head within a phrase, languages can be categorized. A language is head-initial if the head tends to be the first word of a phrase and head-final if the head tends to be at the end. English is an example for a head-initial language while Japanese is an example for a head-final language. Head driven phrase structure grammar (HPSG) parsers can parse a sentence into phrases and mark the heads. Isozaki, Sudoh, et al. (2010) used such a parser to head-finalize the source side in an English to Japanese translation task. In a preprocessing step, they used the parser to recognise the head of every phrase and shift it to the end of the respective phrase. While one might argue that this is essentially hard coding a rule, the used parser still relies on probabilities and the head-finalized source side can still serve as a foundation for more subtle statistical reordering. A combination of head finalization and our approach seems promising, but was discarded for lack of time.

A very promising approach in tree-to-string systems are packed forests as proposed by Mi et al. (2008). While most systems use one parse tree, they use many possible parse trees and store them in a hyper graph. This is basically a tree with additional edges between root and leaves. This hyper tree is used to store several parse trees for the same sentence

In contrast to hyper graphs, Rottmann & Vogel (2007) use a system that stores pos-

sible reorderings in a lattice, which can be seen as a hyper path. Their work is based on Crego & Marino (2006) and uses an aligned parallel corpus with a POS annotated source side. They extract short range rules by searching for crossing alignments between source and target side. Besides, they develop four types of rules, among them two that provide context in form of POS tags/words. This approach has been extended by Niehues & Kolss (2009), who introduce discontinuous rules that allow for reordering over longer distances. Since the decoder translates a reordered test set, the phrase table can be built from the reordered source side of the training set as well. Herrmann et al. (2011) suggest to learn phrases from both, the monotone and the reordered source side. The idea is to enable a better match between test and training data. Some phrases are only observed after reordering. A translation model that extracts phrases in their original order from the corpus would not recognise such a reordered phrase and thus, not know how to translate it. Herrmann et al. (2013) combine both, POS based rules and discontinuous rules, with tree-based rules in order to address many linguistic levels of reordering.
This thesis will examine this work.

4. Modelling Reordering in English to Japanese Translation

4.1 Motivation

After Herrmann et al. could report positive results for the language pairs German-English and German-French, we wanted to try out the used methods on an English-to-Japanese task. German and Japanese are both head-final and, although there are certain liberties, both are SOV languages. English and French on the other hand are both SVO languages and head-initial with some exceptions in French. So it appears to be just a change in reordering direction: Herrmann et al. experimented on SOV to SVO, head-final to head-initial, and we try SVO to SOV, head-initial to head-final. However, there are several additional challenges. The language pairs Herrmann et al. worked with are European languages and quite close whereas English and Japanese are very distant. For instance, the Japanese language uses particles that reflect information such as subject of the sentence or directions. Consequently, Japanese word order is very variable. Only one reference sentence per test sentence,however , can not reflect this properly.

4.2 Used Data

The number of corpora at the beginning of experiments was rather limited, so we took what was available at the time and eventually settled for four corpora:

KFTT The Kyoto Free Translation Task (KFTT) Neubig (2011) is a collection of "Wikipedia articles related to Kyoto", for which a hand made annotation is freely available. This enabled us to evaluate reordering regardless of the quality of other models, so we chose it as test set. Strictly speaking, KFTT adds only a hand crafted alignment (and tokenization) to the *Japanese-English Bilingual Corpus of Wikipedia's Kyoto Articles* released by the National Institute for Information and Communication Technology (NICT). We treat the KFTT as a subset of this corpus that adds alignment information. So even if we only mean the original date without the alignment, we will refer to it as the KFTT to avoid confusion. The corpus itself contains 443849 lines, of which 1235 are used as test set.

TED The TED-corpus is a collection of subtitles from different TED-talks. Cettolo et al. (2012) transformed the data gathered by the TED Open Translation Project into an easy to process corpus. The version we used was segmented by the people at the Nara Institute for Science and Technology (NAIST) and contains 96833 sentences. TED-talks cover a variety of subjects from sciences and arts to personal stories. While the tone is of course restricted due to the occasion of a presentation, TED is a very diverse corpus that is available in many language pairs and was therefore our first choice as a test set. However, since TED does not provide an alignment between English and Japanese words in the subtitles, we eventually opted for KFTT.

Tatoeba The Tatoeba(*jap. example*) corpus stems from tatoeba.org, a site where everyone can submit sentences and translations to sentences. The project works with all language pairs simultaneously by assuming that translations are in a transitive relation. If the system holds a Japanese sentence which has an English translation and the same English sentence is translated into German by a user, the Japanese sentence is automatically mapped to the German one as well. We used a 176976 sentence subset of those translations. As the project relies on 'the crowd' the accuracy and fluency of translations cannot be guaranteed, yet like wikipedia the project has mechanisms to improve the quality of translations. A considerable amount of sentences within the Japanese-English pair of tatoeba stems from the Tanaka corpus[1] added to tatoeba in 2006. The Tanaka corpus is a student crafted Japanese-English corpus whose compilation was overseen by Professor Yasuhito Tanaka at Hyogo University. A description for the methods to obtain the data can be found in Tanaka (2001). During incorporation in WWWJDIC, another online dictionary, prior to incorporation into Tatoeba, the Tanaka corpus was cleansed.

Reiji, Waei Reiji and Waei are partitions of the **Eijiro** corpus. Started by one person to keep track of his vocabulary, it soon increased when he invited friends to contribute. Reiji and Waei sum up to 2428711 sentences.

4.3 System Description

After we have described the used corpora, we now want to give an overview of our baseline system. In a first step, we used GIZA++ (Och & Ney, 2003) to train alignments from all KFTT, TED, Tatoeba, Reiji and Waei. After that, we built several language models with SRILM (Stolcke, 2002) and several phrase tables with Moses. We then ran a series of combinations and chose the one that performed best. For the language model, we compared a model based on the KFTT corpus alone, one based on all five corpora and a model that was a linear combination of all five corpora optimized on the KFTT development set.
For the phrase table, we compared a phrase table over the KFTT corpus and one over all corpora. The evaluation of the resulting six systems showed that the combination of a KFTT only language model and a KFTT only phrase table perform best on the

[1]`http://www.edrdg.org/wiki/index.php/Tanaka_Corpus`

test set from KFTT. Therefore, we used this combination for all further experiments. The final language model contains 3932728 entries, for the phrase table we extracted 4473902 pairs. The reordering rules were extracted from all corpora, unless explicitly stated otherwise. As a decoder, we used our in-house decoder(Vogel, 2003). The resulting translations were evaluated using four different metrics:

BLEU was used because it is widely used and can be seen as a standard in the SMT community.

RIBES was added because it pays more attention to reordering and is popular among researchers on Japanese-English tasks.

τ **and chunk** were added because they focus on alignment alone and cannot be influenced e.g. by a faulty language model. We used the implementation from the lader parser[2] described in Neubig, Watanabe, & Mori (2012). Because we need the alignment rather than the translated sentence, we extracted the alignment by retracing the path in the lattice that was chosen by the decoder. For this path we were able to compute the alignment by looking at the logs from the creation process. Therefore, our alignment given to the τ and chunk metric is the reordering caused solely by our reordering rules. It lacks two aspects of reordering that will take place only later in the decoder: Reordering during decoding time by jumps as exemplified in Section 2.7.4 and reordering within phrases mentioned in Section 2.7.1.

4.4 Reordering Experiments

Herrmann et al. (2013) were able to improve over POS based rules and discontinuous rules with the use of treerules for German to English and German to French translation. The goal of this thesis is to evaluate treerules on an English-to-Japanese task. In order to achieve this, we analysed the influence of different reordering methods to translation quality.
Like Herrmann et al., we use a reordering step prior to decoding in which the original sentence is reordered. The found permutations and the original sentence are stored in a lattice structure.

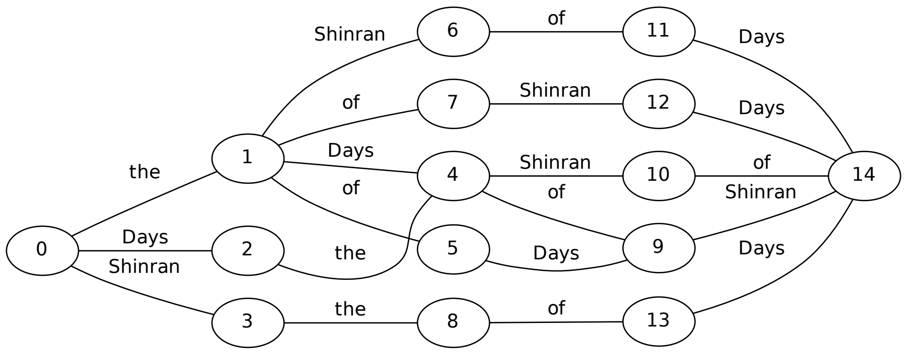

Figure 4.1: An example of a lattice that stores our reorderings.

[2]http://phontron.com/lader/

A lattice is a partially ordered set with two special elements: A unique upper bound and a unique lower bound. We store our tokens on the edges in the lattice. Figure 4.1 shows an instance of such a lattice. The numbers on the nodes depict the order: 14 is the upper bound, 0 the lower bound.

We will now give an overview of the essential experiments. First, we built a system that uses no preordering(*plain*) as described in 4.3. This baseline system was extended by several approaches as illustrated in the following sections.

4.4.1 Short-Range Reordering Rules

First, we extended the *plain* system by adding the POS-based reordering rules described in Rottmann & Vogel (2007) who based theirs on Crego & Marino (2006). To obtain these rules, an aligned parallel corpus is needed where the source side is annotated with POS tags. Whenever we encounter two source words with crossed target words, we check for a rule. Formally speaking, we look for two source words at positions $i < j$ with aligned target words at $a_i > a_j$. When found, we can extract a sequence of POS tags. Rottmann & Vogel also allow a context of one or two tags or words either preceding or following the extracted sequence. Also they allow the sequence to be in form of plain words but in that case no context is used. For a rule to be saved, a threshold of five occurrences in the corpus has to be reached. For accepted rules a frequency is calculated based on the number of times such a pattern occurred and the number of times the rule was extracted in such a situation. These rules can handle local reorderings like the peculiarity of the French language to have the adjective follow the noun, as in *"an interesting book"* and *"un livre intéressant"*. We were able to extract an overall of 689333 short range rules. Adding them to the existing *plain* system resulted in the *+short* system.
However, as a matter of fact, the extracted rules are of short range (on average 4.61 tokens in our case) and since we only look at sequences, long-range reorderings, where the 2 sequences in question are non-consecutive, cannot be modelled properly.

4.4.2 Long-Range Reordering Rules

The next system(*+long*) added discontinuous rules as depicted by Niehues & Kolss (2009). Discontinuous rules were introduced in an effort to tackle long-range reorderings. They allow the translation system to perform long-range reordering without the need to look at words in between.

"Das wird mit derart unterschiedlichen Mitgliedern unmöglich sein."

"That will be impossible with such disparate members."

Figure 4.2: An example for a long-range rule.

As an example, they give the German sentence and its English translation in Figure 4.2: *'wird'* and *'sein'* translate to the English words *'will'* and *'be'*. But while *'wird'* and *'sein'* enclose the object, the corresponding English counterparts stand together. Such a reordering cannot be modelled with shortrules since they rely on an explicit sequence to reorder and cannot abstract a subsequence. A rule such as:

$MD\ *\ VB \rightarrow MD\ VB\ *$ however could model the reordering correctly, where MD matches ‘will’ and VB matches ‘be’. A depiction of all four rules introduced by Niehues & Kolss (2009) can be seen in Figure 4.3.

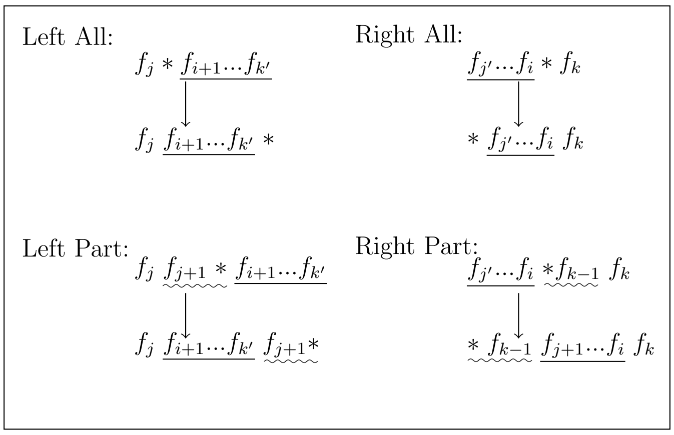
Left All:
$f_j * \underline{f_{i+1} \cdots f_{k'}}$
$f_j\ \underline{f_{i+1} \cdots f_{k'}} *$
Right All:
$\underline{f_{j'} \cdots f_i} * f_k$
$* \underline{f_{j'} \cdots f_i}\ f_k$
Left Part:
$f_j\ f_{j+1} * \underline{f_{i+1} \cdots f_{k'}}$
$f_j\ \underline{f_{i+1} \cdots f_{k'}}\ f_{j+1} *$
Right Part:
$\underline{f_{j'} \cdots f_i} * f_{k-1}\ f_k$
$* f_{k-1}\ \underline{f_{j+1} \cdots f_i}\ f_k$

Figure 4.3: Discontinuous rules: f_j and f_k are the words before/after the reordering. f_j' and f_k' are the first aligned word after f_j and the last aligned word before f_k respectively.

Here, we could extract 433189 rules.

4.4.3 Tree-Based Reordering Rules

These systems were followed by a system that adds the treerules described by Herrmann et al. (2013)(*+tree*). We will describe shortly these rules. So far we have looked at reordering on the basis of sequences and ignored the structure of the sentence. To handle reordering on a constituent level, we use reordering rules based on parse trees as outlined by Herrmann et al. (2013). Since they operate on parse trees, in addition to the requirements stated in 4.4.1, now a parse tree for each sentence on the source side is required. To extract the rules, the parse tree of any sentence on the source side is traversed top-down and if a subtree contains a crossing alignment, a rule is extracted.

An example for a rule extraction is shown in Figure 4.4. A reordering takes place between the constituents *VVPP* and *NP*, hence, the rule *VP PTNEG NP VVPP* $\rightarrow$ *VP PTNEG VVPP NP* is extracted. The first POS tag in the rule, *VP*, denotes the head constituent of the respective subtree, the following tags describe the child nodes. For this system additional 11506 rules were obtained.

4.4.4 Partial Tree Based Rules

Working on top of those 3 systems, we built a system that added partial application of treerules as described in the same paper. We will refer to this system as *+partial.* For treerules the head of a subtree and all its child nodes are extracted. The idea of partial treerules is that a treerule does not necessarily depend on all child nodes. So

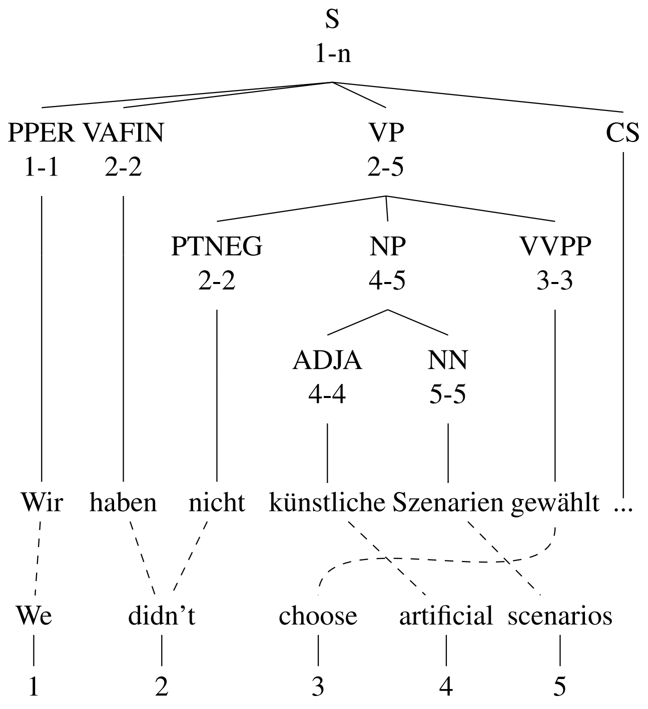

Figure 4.4: A sample parse tree, source Herrmann et al. (2013)

for a subtree the head and every continuous sequence of child nodes are extracted. This system added another 10752 rules.

In addition, we evaluated different variants of the reordering approach using tree-based rules. Beside the basis tree reordering, we also restricted the source for reordering rules, applied rules recursively, worked with lattice phrase extraction and used a different parsing approach. Those experiments are described in more detail later on. As in the paper by Herrmann et al., the order of systems is inclusive, i.e. *+tree* adds treerules on the lattice produced by *+long*. We will shortly discuss the reasons for this to avoid confusion. The decision to built this series of experiments in a cumulative manner causes erroneous reordering in *+short* to prevail into later systems and act as noise for the decoder in all following systems. System *long* for instance might even perform better if examined isolated. But as Herrmann et al. argue, the different methods address distinct linguistic levels, so pairwise comparing the different approaches isolated gives us little information. Comparing how much a new technique can add to an existing system however, gives us valuable information about what we actually gain. In the following, we will describe the afore mentioned variations to the *+tree* system.

4.4.5 Restricting Rules to KFTT

Since the Japanese grammar allows for a wide variety of correct word orders but our test set provides only one reference per hypothesis, we wanted to match the style of word order used in the test set better. In order to do so, we restricted the extraction of rules to the development set of the KFTT corpus from which our test

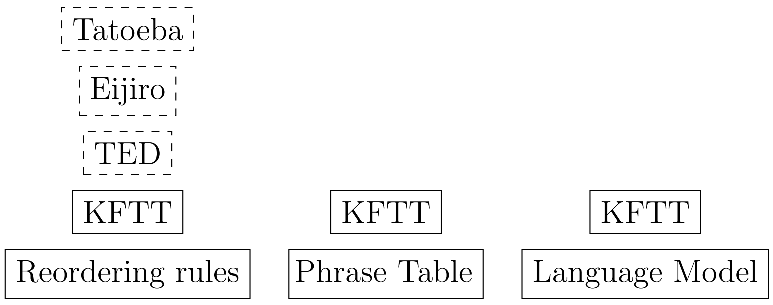

Figure 4.5: Juxtaposition of the corpora used for different components. For better adaption TED, Eijiro and Tatoeba (dashed) are removed from the reordering rules.

set is derived. A depiction of the components can be found in Figure 4.5. This restriction left us with 547911 rules, about half as much as in Section 4.4.

4.4.6 Recursive Rule Application

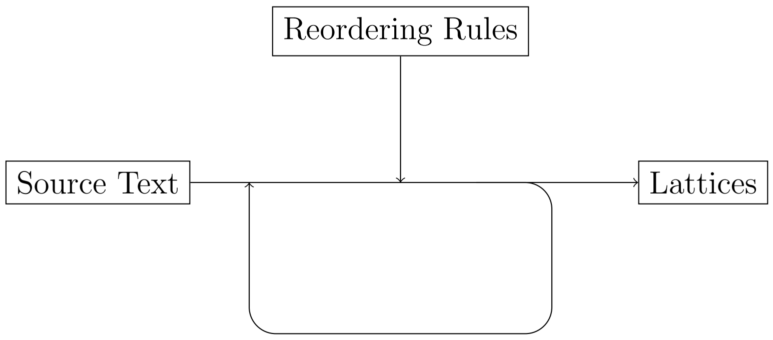

Figure 4.6: Schema of the concept of recursive rule application.

The correct reordering of a sentence cannot necessarily be described with just one rule. To handle such cases, where a grammatical rule needs two or more tree-rules, we allowed this system to apply rules not only on the original source side sentence but also on an already reordered sentence. The set-up is depicted in Figure 4.6. To avoid infinite loops, once reordered constituents are excluded from additional reordering.

4.4.7 Examination of Long Range Reordering

Niehues & Kolss (2009) present two types of rules: right rules, where the anchor tag is on the right side of the reordering, and left rules, where the anchor is on the left side of the reordering. While it is intuitive to use both types to extract as exhaustively as possible, we expected that one type will dominate. Therefore, we performed an additional experiment and built a *+left* and a *+right* system to compare with the *+long* system. Niehues & Kolss used left rules for German to English and German to French. As we argued in 4.1, English to Japanese can be seen as an opposite translation direction looking at subject, object and verb order, and head movement. So we expect better results for the right rules. Surprisingly, we could extract more left rules than right rules. We see this as an indication that the right rules generalise better, thus a smaller number is needed to express the same phenomena.

4.4.8 Lattice Phrase Extraction

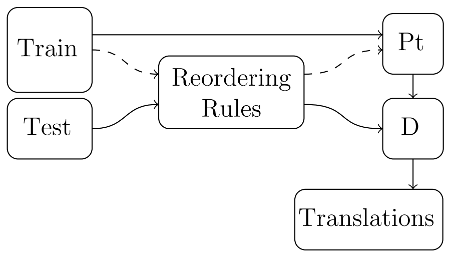

Figure 4.7: Schema of the changes made through LPE. The dashed connection is added.

As we decode reordered lattices we expected an improvement when using a phrase table trained on a reordered source side. In order to obtain such a phrase table, we extended the existing phrase table by phrases extracted from the lattices of the reordered source side of the training corpus. Figure 4.7 shows the different setup of the translation system. Like the previous one, this experiment uses only POS and discontinuous reordering rules.

4.4.9 Approaches to Parsing

In Section 4.4.3, we mentioned that we use parse trees for our tree-based rules. These parse trees are obtained through a parser that assigns POS-tags to every word and also sorts the single words into a tree structure. We use two approaches in this thesis, which we will briefly explain.

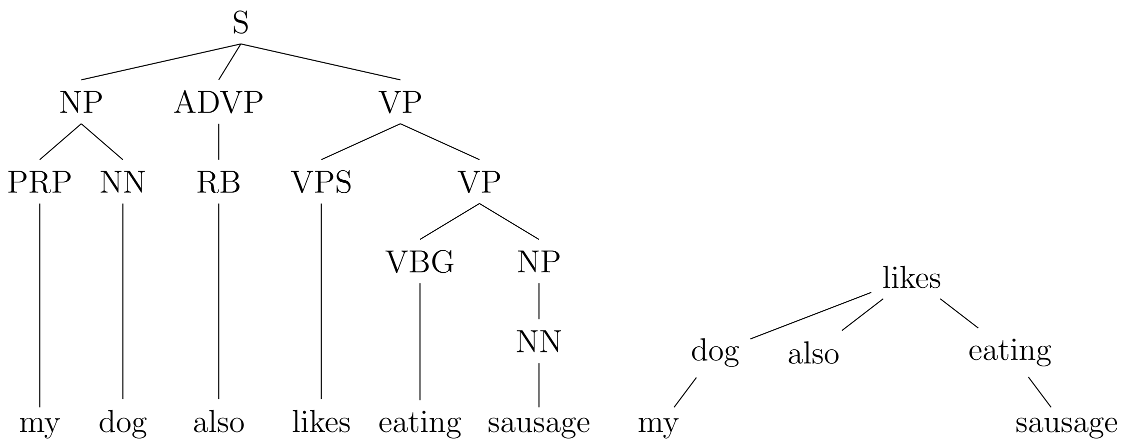

Figure 4.8: The same sentence parsed as constituents on the left and as dependencies on the right

4.4.9.1 Constituent Parsing

In constituent parsing words are grouped into linguistic phrases, which are grouped into other phrases until one phrase characterises the whole sentence. The tree on the left in Figure 4.8 shows a constituent tree. Nagao (1981) explicitly disproves of constituency because of its free word order, so we considered alternatives.

4.4.9.2 Dependency Parsing

In dependency parsing, no additional tokens are introduced. Every word, except the root word is ‘dependent’ on another word directly. An example for a dependency parse tree can be seen on the right side in Figure 4.8. POS information is not shown. While some authors view dependencies as a data structure opposed to trees, we do treat them as trees to fit our purpose. To compare both approaches, we executed all experiments again using dependency parse trees for the rule creation.

4.5 Manual Analysis

The Japanese language allows for a very flexible word order, but there are two basic requirements: the subject must be at the beginning of the sentence and the verb at the end. While these conditions influence all four automatic metrics, there is no way to identify the extend to which any condition alone is satisfied. To see to what extend the second condition is satisfied, we did a manual evaluation of 100 lattices for the elementary configurations where we successively added new rule types. We analysed two questions:

1. Is there a path in the lattice which has the verb at the end?
2. Is such a path chosen as hypothesis by the decoder?

As some verb constructs such as *“be called”* consist of two words, we made an exception for them: As long as a verb and its auxiliary occupied the last two positions, we did not mind if the verb was only at the second to last position. It seems possible to automate this behaviour, but for a lack of time we decided against it.

5. Results

In this chapter, we present the results for the experiments illustrated in the previous Chapter 4. To eliminate statistical anomalies, every system was run five times. The values presented in the tables are averages of these five scores. All experiments use a subset of the KFTT corpus as a test set.

5.1 Constituent parsing

metric	**plain**	**+short**	**+long**	**+tree**	**+partial**
BLEU	14.77	16.29	16.22	16.51	16.49
RIBES	62.88	64.54	64.77	65.44	65.48
chunk	63.094	62.918	61.226	61.107	61.115
τ	78.154	78.893	78.802	78.757	78.781

Table 5.1: Scores for syntax trees

Table 5.1 shows our results for reordering experiments without any extensions. Looking at the RIBES scores, we see the expected increase over more exhaustive methods. On BLEU we also see an overall improvement compared to the plain system and POS-only rules. The chunk values are all worse. τ values on the other hand can all improve.

5.1.1 Restricting Rules to KFTT

In the Japanese language, word order is relatively free. Therefore, in this experiment we restricted rule extraction solely to the KFTT corpus.

The results for this are shown in Table 5.2. While the overall BLEU and RIBES scores are lower, especially for the treerules, we get a monotonous increase. Chunk and τ improve for treerules. Chunk can additionally improve for discontinuous rules, τ for partial.

	all rules				**kyotorules**			
metric	**+short**	**+long**	**+tree**	**+partial**	**+short**	**+long**	**+tree**	**+partial**
BLEU	16.29	16.22	16.51	16.49	16.07	16.21	16.26	16.33
RIBES	64.54	64.77	65.44	65.48	64.37	64.76	64.81	64.98
chunk	62.918	61.226	61.107	61.115	62.669	61.534	61.412	60.066
τ	78.893	78.802	78.757	78.781	78.723	78.756	78.803	79.816

Table 5.2: Scores for syntax trees with rule extraction restricted to the Kyoto corpus

5.1.2 Recursive Rule Application

We will now look at the results for our systems where we allowed reordering rules to be executed several times on the same path. The scores for recursive rule application

	non-recursive		**recursive**	
metric	**+tree**	**+partial**	**+tree**	**+partial**
BLEU	16.51	16.49	16.72	16.67
RIBES	65.44	65.48	65.59	65.6
chunk	61.107	61.115	61.683	61.038
τ	78.757	78.781	80.089	78.936

Table 5.3: Recursive Rule Application

are depicted in Table 5.3. As expected, in BLEU and RIBES score, we can see that the systems outperform the plain system as well as the non-recursive treerules and partial systems: The BLEU score increases by 0.21 and 0.18 while RIBES increases by 0.15 and 0.12. τ can improve over both, *treerules* and *partial*, and chunk increases clearly over the non-recursive treerules.

5.2 Dependency Trees

In this section we present the results for configurations where we exchanged constituency parsing for dependency parsing. For comparison we show the results for shortrules and longrules as well. Please keep in mind that they are the same results as for constituency parsing since POS rules do not rely on a parser.

	syntax		**dependency**	
metric	**+tree**	**+partial**	**+tree**	**+partial**
BLEU	16.51	16.49	16.5	16.46
RIBES	65.44	65.48	65.03	65.01
chunk	61.107	61.115	61.319	61.247
τ	78.757	78.781	78.712	78.848

Table 5.4: Scores for dependency trees

In Table 5.4 the scores for the standard line of experiments based on dependency parsing are listed. While still outperforming the plain system, we could not improve over tree-based rules parsed with constituency trees. BLEU shows a decrease of 0.01 and 0.03, RIBES decreases by 0.41 and 0.47. A higher decrease in RIBES is usual as the metric punishes reordering explicitly. The decrease itself may be explained by the fact that the rules where developed for constituency trees. An exhaustive examination of all parameters used in extraction and application of the rules, e.g. the pruning threshold, may lead to better results. However, Chunk and τ do improve, save for the treerules in τ.

5.2.1 Restricting Reordering Rules to KFTT

			constituency		**dependency**	
metric	**+short**	**+long**	**+tree**	**+partial**	**+tree**	**+partial**
BLEU	16.07	16.21	16.26	16.33	16.24	16.24
RIBES	64.37	64.76	64.81	64.98	64.77	64.82
chunk	62.669	61.534	61.412	60.066	61.544	61.718
τ	78.723	78.756	78.803	79.816	78.745	78.547

Table 5.5: Scores for dependency trees with rule extraction restricted to the Kyoto corpus

Table 5.5 depicts the scores for reordering rules solely based on the KFTT corpus. Compared to the constituency parsed rules, we see a decrease again. However, if we compare the relative decrease to the values for the systems with rules from all corpora in Table 5.4, the decrease in BLEU points is roughly the same while the decrease in RIBES is smaller: The decrease on the constituency side is 0.63 and 0.5 while on the dependency side we have 0.26 and 0.19. For chunk, the scores decrease, except for *+tree*. τ decreases overall. Compared to Table 5.4 however, we see an increase in both chunk values and in *+tree* for τ.

5.2.2 Recursive Rule Application

	constituency				**dependency**			
	non-recursive		**recursive**		**non-recursive**		**recursive**	
metric	**+tree**	**+part**	**+tree**	**+part**	**+tree**	**+part**	**+tree**	**+part**
BLEU	16.51	16.49	16.72	16.67	16.5	16.46	16.62	16.58
RIBES	65.44	65.48	65.59	65.6	65.03	65.01	65.41	65.13
chunk	61.107	61.115	61.683	61.038	61.319	61.247	61.285	61.085
τ	78.757	78.781	80.089	78.936	78.712	78.848	78.712	78.831

Table 5.6: Recursive Rule Application

Depicted in Table 5.6 are the scores for recursive rule application with rules on a dependency basis. As expected, recursive rule application can again outperform the non-recursive system in BLEU and RIBES. The values for chunk both decrease. For τ *+tree* remains unchanged and in partial we have a slight decrease. The gain over the non-recursive system in BLEU cannot improve compared to constituency parsing. On constituency based parsing the scores improved by 0.21 and 0.18 (Table 5.3), on dependency we have 0.12 each. Looking at chunk and τ the improvement in the constituent table is bigger as well. Interestingly, RIBES can improve more than for constituency trees: On constituency we had an improvement of 0.15 and 0.12, each over the standard line, on dependency we have 0.38 on treerules but only 0.12 on partial rules.

5.3 Closer Examination of Long Range Rules

As we mentioned in Section 4.4.2, there are two different types of rules that make up the discontinuous rules: left rules, which have an anchor that is not shifted on the left part of the pattern, and right rules, which have the anchor on the right.

metric	**+left**	**+right**	**+both**
BLEU	16.45	16.44	16.22
RIBES	64.25	64.79	64.77
chunk	61.345	61.46	61.226
τ	78.145	78.97	78.802

Table 5.7: Results for the comparison of discontinuous rules.

Table 5.7 shows the results of the comparison. The *+both* system is identical to *+long* and was merely renamed to fit in the context. While left rules get a slightly higher BLEU score, for RIBES, chunk and τ right rules perform better. This confirms our hypothesis in 4.4.7 that right rules would outperform left rules. Interestingly, choosing only one type yields mostly better results than the combination. Right rules alone can outperform the combination of both rules on every metric. For left rules BLEU and chunk give better results than the combination while RIBES and τ are considerably weaker.

5.4 Lattice Phrase Extraction

metric	**+left**	**+right**	**+both**
BLEU	16.25	16.53	16.4
RIBES	64.17	65.09	64.91
chunk	61.595	61.321	61.081
τ	77.603	79.042	78.772

Table 5.8: Results for LPE

Table 5.8 shows the scores for lattice phrase extraction (LPE), where we worked with a phrase table built from the reordered source side. We compared again with only left and right rules, respectively. Compared with non-LPE, we see gains in BLEU and RIBES for the combination of both rules. For left rules alone, we see a decrease except for chunk. For right rules alone, we see an increase in scores except for chunk. If we compare the LPE systems between each other, we see again that

+right is leading in scores. Among non-LPE systems *+right* was outperformed only by *+left* in the BLEU metric by 0.01 points, now it is only outperformed by *+left* in the chunk metric by 0.274.

5.5 Manual analysis

condition	**plain**	**+short**	**+long**	**+tree**	**+partial**
in Lattice	-	24	68	85	85
Taken	6	12	19.8	25.8	25.8

Table 5.9: Manual analysis. Since the *plain* configuration does not use lattices the respective entry is empty.

We will now have a look at the scores from the manual evaluation of the experiments without extensions. For each system we evaluated 100 sentences for which the reference translation definitely required the verb at the end of the sentence. As with the previous automatically scored systems, we worked with five runs per system. For every run we counted the number of lattices that contain a path which has the correct verb shifted to the end. After that, we determined for every lattice whether such a path was chosen by the decoder. Then we averaged over the five runs per system. Table 5.9 shows the results. Depicted in the first row is the percentage of hypotheses that have a path in the lattice which has the correct verb at the end. In the second row we listed the percentage of hypotheses where such a path was chosen. We notice a strong increase in paths that offer a hypothesis with the correct verb at the end, especially from short to long. This is expected behaviour as *longrules* are the first to offer word shifting over a distance, which is necessary for the English-Japanese language pair. Please note that while many lattices contain paths with some verb at the end, we required the correct one which might explain that the *treerules* can get only 85%. Though, we could not analyse every system manually due to time constraints, since most of our systems extend the *+partial* system, we can assume them to offer at least 85% lattices with such a valid path in the lattice. The percentage of cases where such a path was chosen is of course lower. While the constraint to have the verb at the end is successively satisfied, the numbers stay very low. This could be improved by having the decoder to pay special regard to this issue. An extra pruning step filtering out paths that violate this constraint or adding an extra feature to the decoder, which tests if the last word of a hypothesis is a verb, might point the decoder in the right direction. However, such a step would come very close to hardcoding information about the language when the main goal should be to find generic algorithms and models to solve the problem.

5.6 Discussion

We will shortly discuss the implications of the results presented above.
In BLEU and RIBES, we have seen a mostly monotone increase by addressing more grammatical layers through the introduction of POS-based, discontinuous and tree-based rules. This confirms our hopes that adding these rules can improve the scores in conventional metrics.

In an attempt to adapt more closely to the test set, we restricted the source of rule extraction from all corpora to KFTT. However, we got lower scores, indicating that it is in fact desirable to extract as many rules as possible.
Herrmann et al. (2013) suggest to use rules multiple times by applying reordering rules to already reordered source sentences in order to express more complex rules. This conjecture is supported by our findings.
To get a better grip at the syntax, we experimented with dependencies as an alternative grammatical structure. We can see no clear advantage of choosing dependencies over constituents: For the *plain* system as well as the two variations, the BLEU and RIBES scores slightly decrease and the reordering based scores slightly increase in some cases. Therefore, on the basis of our experiments, we cannot recommend dependency parsing. We mentioned in 5.2 that this might be due to the fact that the reordering rules we use were developed with constituency trees in mind. An adjustment to their extraction process might improve the scores.
The discontinuous rules we use in the *+long* system have two types of rules. To better adapt to the language pair we ran systems with only one type each, expecting the system with right rules only to perform better. This could be confirmed by our data. The right rules alone could outperform the combination on every metric. For lattice phrase extraction, where we built the phrase table from a reordered source side, the right rules only system was able to outperform the combination again. The improvement over the left only system is even clearer here.
In our manual analysis we examined whether the condition to have the verb at the end of the sentence was satisfied or not. We see that while in the best system only every fourth eventually chosen translation hypothesis has the verb at the end, 85% of the lattices contain such a hypothesis. We can deduce that our reordering rules capture this rule quite well. Since the decoder can see whether his currently chosen path in the lattice complies with the rule only at a very late state, we suggest a pruning step prior to decoding to filter such noisy paths out.

Another point are the rather disappointing values for chunk and τ. Both metrics were introduced in the hope to give a deeper insight to the reordering that would not be tainted by the translation choices for example. Accordingly, we expected that improvement for the lexical metrics BLEU and RIBES would always imply an even clearer improvement for chunk and τ which look at reordering alone. Especially, because our goal was to improve translation quality specifically by tackling reordering. For chunk however, the *plain* system outperforms every experiment we conducted. Despite extensive efforts, we were unable to find a satisfying answer for this in the given period of time. We did debug the software we used for evaluation and were able to find and resolve an error in our toolchain but the chunk scores remain low. We can only suggest the following explanations for this behaviour: One idea is that we optimise our systems on BLEU, so other metrics have a bit weaker scores. Also, one could argue that an essential part of reordering happens in the decoder. In Section 4.3, we noted that the alignment we score is the one before decoding, therefore lacking reordering during decoding and within phrases. This could be an explanation for the relatively low scores for chunk. Additionally, as we already mentioned, our target language has a very free word order but we have only one reference sentence available, so our hypotheses might be better than they score.

6. Conclusion

Reordering is an important issue in statistical machine translation. Statistical machine translation systems struggle to provide a reordering mechanism which is able to abstract from observed example sentences to general rules. Among many proposed solutions, this thesis focuses on preordering combined with a small reordering window in the decoder. We extract reordering rules from the plain sentences as well as tagged parse trees in order to address three word levels. The source sentence in original order is stored in a lattice along with all permutations according to the extracted rules. The resulting lattice is processed by a decoder with a small reordering window. This approach has been tested successfully on the language pairs German-English that displays similar grammatical characteristics as Japanese-English. While English is an SVO (subject-verb-object) language, Japanese and German can be seen as SOV languages. This means we have to shift the verb across the whole sentence. We conducted experiments on English to Japanese translation using part-of-speech and tree-based reordering. We compared the translations produced by a system using POS based reordering with the translations produced by a system using both POS and tree-based reordering. Furthermore, the benefits of partial rules and recursive rule application were investigated. In addition to that, we performed a deeper analysis of discontinuous reordering rules and compared these rules with a system that obtained phrase table entries from a reordered training set. To evaluate the overall translation quality of the different systems we used BLEU and RIBES. For the evaluation of the reordering alone we used Kendall's τ, a rank based metric along with *'chunk'* which looks at consecutive word sequences. Besides, we performed a manual analysis to one critical condition: Is the verb always shifted to the end of the sentence?
We could show that discontinuous rules help to put the verb at the end of the sentence by automatic and manual evaluation. Additionally, we have shown that our system performs better with constituency based parse trees than with dependency parsed trees. We could also report that learning rules from a broad domain increases quality. With our experiments with recursive application of reordering rules, we were able to reproduce the finding of Herrmann et al. (2013) that recursive rule application outperforms non-recursive application. We can further report that lattice phrase extraction can improve the scores, as does the refinement of discontinuous

rules. Our manual analysis showed that a system with the reordering methods described in Herrmann et al. (2013) alone, offers lattices that provide a path where the verb gets shifted to the end of the sentence 85% of the time.

6.1 Future Work

There are several possible extensions to the experiments we presented. Future work will include testing out combinations with other approaches such as reordering on an already head finalized source side or comparison with an other parser, i.e. an HPSG parser, or combination of parsed lattices.

While we could not report any reason for using dependency parsing, it might be worthwhile to use it in the translation direction Japanese-to-English as the idea to capture Japanese grammar with dependencies sounds very appealing.

Also, the afore mentioned automatic evaluation of where the verb is shifted would be a possibility to refine the analysis. Using the POS-tags assigned by the parser, we could filter out sentences without a verb in both source and reference and at the last position in the reference. From the remaining sentence pairs, the correct verb in the source sentence could be identified using the gold alignment from KFTT. Then, we could automatically check whether this token appears at a late position in the lattice and whether the chosen path has the verb at the last position. For this thesis, we decided against it for lack of time to develop such a tool but with enough time, it would allow us to test the condition on every experiment.

Bibliography

Cettolo, M., Girardi, C., & Federico, M. (2012, May). Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th conference of the european association for machine translation (eamt)* (pp. 261–268). Trento, Italy.

Chiang, D. (2005, June). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd annual meeting of the association for computational linguistics* (pp. 263–270). Ann Arbor, Michigan: Association for Computational Linguistics.

Crego, J. M., & Marino, J. B. (2006, December). Reordering experiments for n-gram-based smt. In *Spoken language technology workshop* (pp. 242–245). Palm Beach, Aruba.

Herrmann, T., Mediani, M., Niehues, J., & Waibel, A. (2011, July). The karlsruhe institute of technology translation systems for the wmt 2011. In *Proceedings of the sixth workshop on statistical machine translation* (pp. 379–385). Edinburgh, Scotland: Association for Computational Linguistics.

Herrmann, T., Niehues, J., & Waibel, A. (2013, June). Combining word reordering methods on different linguistic abstraction levels for statistical machine translation. In *Proceedings of the seventh workshop on syntax, semantics and structure in statistical translation* (pp. 39–47). Atlanta, Georgia: Association for Computational Linguistics.

Isozaki, H., Hirao, T., Duh, K., Sudoh, K., & Tsukada, H. (2010, October). Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 944–952). Cambridge, MA: Association for Computational Linguistics.

Isozaki, H., Sudoh, K., Tsukada, H., & Duh, K. (2010, July). Head finalization: A simple reordering rule for sov languages. In *Proceedings of the joint fifth workshop on statistical machine translation and metricsmatr* (pp. 244–251). Uppsala, Sweden: Association for Computational Linguistics.

Kendall, M. G. (1938, June). A new measure of rank correlation. *Biometrika*, *30*(1/2), 81–93.

Koehn, P. (2010). *Statistical machine translation* (1st ed.). New York, NY, USA: Cambridge University Press.

Liu, Y., Liu, Q., & Lin, S. (2006, July). Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st international conference on* *computional linguistics and 44th annual meeting of the association for computational linguistics* (pp. 609–616). Sydney, Australia: Association for Computational Linguistics.

Mi, H., Huang, L., & Liu, Q. (2008, June). Forest-based translation. In *Proceedings of the 46th annual meeting of the association for computational linguistics: Human language technologies* (pp. 192–199). Columbus, Ohio: Association for Computational Linguistics.

Nagao, M. (1981, October). A framework of a mechanical translation between japanese and english by analogy principle. In *Artifiical and human intelligence: Edited review papers presented at the international nato symposium on artificial and human intelligence* (pp. 173–180). Lyon, France.

Neubig, G. (2011). *The Kyoto free translation task.* http://www.phontron.com/kftt.

Neubig, G., Watanabe, T., & Mori, S. (2012, July). *Inducing a discriminative parser to optimize machine translation reordering.* Jeju Island, Korea.

Neubig, G., Watanabe, T., Mori, S., & Kawahara, T. (2012, July). Machine translation without words through substring alignment. In *The 50th annual meeting of the association for computational linguistics* (pp. 165–174). Jeju, Korea: Association for Computational Linguistics.

Niehues, J., & Kolss, M. (2009, March). A pos-based model for long-range reorderings in smt. In *Proceedings of the fourth workshop on statistical machine translation* (pp. 206–214). Stroudsburg, PA, USA: Association for Computational Linguistics.

Och, F. J. (2003, July). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st annual meeting on association for computational linguistics - volume 1* (pp. 160–167). Stroudsburg, PA, USA: Association for Computational Linguistics.

Och, F. J., & Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, *29*(1), 19–51.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-j. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 311–318). Philadelphia, Pennsylvania: Association for Computational Linguistics.

Quirk, C., Menezes, A., & Cherry, C. (2005, June). Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of 43rd annual meeting of the association for computational linguistics* (pp. 271–279). Ann Arbor, Michigan: Association for Computational Linguistics.

Rottmann, K., & Vogel, S. (2007, September). Word reordering in statistical machine translation with a pos-based distortion model. In *Proceedings of the 11th international conference on theoretical and methodological issues in machine translation* (pp. 171–180).

Stolcke, A. (2002). Srilm – an extensible language modeling toolkit. In *In proceedings of the 7th international conference on spoken language processing (icslp 2002* (pp. 901–904).

Sumita, E., Iida, H., & Kohyama, H. (1990, June). Translating with examples: a new approach to machine translation. In *The third international conference on theoretical and methodological issues in machine translation of natural language* (pp. 203–212). Austin, Texas.

Tanaka, Y. (2001, September). Compilation of a multilingual parallel corpus. In *Proceedings of pacling 2001* (pp. 265–268).

Vogel, S. (2003, October). Smt decoder dissected: Word reordering. In *Int. conf. on natural language processing and knowledge engineering (nlp-ke)* (pp. 561–566). Beijing, China.