Dependency Reordering Features for Japanese-English Phrase-Based Translation

by

Jason Edward Katz-Brown

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2008

Author .....
Department of Electrical Engineering and Computer Science
August 22, 2008

Certified by .....
Michael Collins
Associate Professor of Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

Dependency Reordering Features for Japanese-English Phrase-Based Translation

by

Jason Edward Katz-Brown

Submitted to the Department of Electrical Engineering and Computer Science
on August 22, 2008, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Translating Japanese into English is very challenging because of the vast difference in word order between the two languages. For example, the main verb is always at the very end of a Japanese sentence, whereas it comes near the beginning of an English sentence. In this thesis, we develop a Japanese-to-English translation system capable of performing the long-distance reordering necessary to fluently translate Japanese into English. Our system uses novel feature functions, based on a dependency parse of the input Japanese sentence, which identify candidate translations that put dependency relationships into correct English order. For example, one feature identifies translations that put verbs before their objects. The weights for these feature functions are discriminatively trained, and so can be used for any language pair. In our Japanese-to-English system, they improve the BLEU score from 27.96 to 28.54, and we show clear improvements in subjective quality.

We also experiment with a well-known technique of training the translation system on a Japanese training corpus that has been reordered into an English-like word order. Impressive results can be achieved by naively reordering each Japanese sentence into reverse order. Translating these reversed sentences with the dependency-parse-based feature functions gives further improvement.

Finally, we evaluate our translation systems with human judgment, BLEU score, and METEOR score. We compare these metrics on corpus and sentence level and examine how well they capture improvements in translation word order.

Thesis Supervisor: Michael Collins
Title: Associate Professor of Computer Science

Acknowledgments

Michael Collins was a wonderful advisor, teacher, and role model as an excellently cool researcher and person.

John Guttag also offered unflaggingly solid advice throughout my time at MIT.

Ignacio Thayer and Franz Och taught me patiently and hugeously about statistical machine translation and convinced me that I could contribute, which I was sure was impossible after being stonewalled by Brown *et al.* [1993] my first week.

I am historically beholden to the truly enjoyable and uppropping company in Mike's Group, Google Translate Group, NTT Machine Translation Research Group, and all my friends in between.

Most majorly, 心から thank you to Clair, Richard, and Daniel for their awesome omnimode support at every time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Japanese sentences have vastly different anatomy compared to English sentences. For example, the main verb of a Japanese sentence always comes at the end of the sentence, whereas it comes near the beginning of an English sentence. It follows that to translate a Japanese sentence into English, one must prolifically and accurately reorder the Japanese words to get a fluent English translation. In this thesis, we built a machine translation system that can learn to do this reordering between Japanese and English sentences accurately, using a novel technique that can be applied to translation between any language pair.

Our technique is to translate a **dependency graph** of the Japanese sentence with a phrase-based translation system. This dependency graph tells us how the Japanese words relate to each other. Our translator uses this dependency analysis to reorder the Japanese words during translation and produce English translations that have key dependency relations in the correct order. For example, it is critical that active English verbs come before their object. The system automatically learns to perform the long-distance reordering of a sentence-final Japanese verb to before its object.

Many Japanese→English machine translation systems, such as Yahoo Babel Fish [Yahoo, 2008], rely on hand-built grammars and reordering rules, which are costly to assemble and update. Recent systems take a wholly statistical approach, requiring only a large corpus of parallel text for training. However, these systems perform long-distance word reordering neither efficiently nor accurately. This thesis contributes a

powerful long-distance reordering model to today's best statistical machine translation systems. In Chapter 2, we review where our work fits into the landscape of previous work on statistical reordering models.

We introduce two methods of incorporating Japanese dependency analysis into a state-of-the-art Japanese→English machine translation system to improve translation quality. The first method is to reorder the Japanese training corpus into an English-like word order before training, as Wang et al. [Wang *et al.*, 2007] showed to be effectual for Chinese translation. The second method is to add feature functions that identify translations in which certain dependency relations are translated in the correct order. Used together, these methods improved BLEU score 27.96→28.74 on the test corpus used in the NTCIR-7 Patent Translation Task [Fujii *et al.*, 2007]. We explicate these two methods in Chapters 3 and 4.

Several automatic metrics like BLEU score have been developed to automatically compare the quality of machine translation systems, but their ability to capture differences in word order is suspect [Callison-Burch *et al.*, 2006]. In Chapter 5, we present detailed analysis of the results of our orchestra of experiments. We compare three measures of translation quality: human evaluation, BLEU score [Papineni *et al.*, 2001], and METEOR fragmentation score [Lavie and Agarwal, 2007]. We show that despite its lack of an explicit reordering metric, in practice BLEU score is useful for evaluating systematic differences in word order.

In Chapter 6, we outline future work and reframe the contributions of this thesis.

Let us start by briefly looking at the challenges of translating Japanese into English and previewing for how this thesis will tackle them. Section 1.1 introduces Japanese grammar, and Section 1.2 gives background on current approaches to Japanese→English machine translation. Section 1.3 looks at problems with existing systems, and we finish up with an overview of how this thesis improves the state of the art: Section 1.4 introduces novel features that integrate dependency analysis into a phrase-based translation system and Section 1.5 shows how reordering Japanese sentences into English word order before translating can also improve translation quality.

1.1 Japanese grammatical challenges

The word order of Japanese is very different from that of English. Two features of Japanese grammar account for many of the differences that make Japanese machine translation challenging. First, the verb comes at the end of the sentence, as in this example.

(1.1) 先生が　　お茶を 飲みました 。
Teacher-*Nom* tea-*Acc* drank .
"The teacher drank tea."

The verb '飲みました', "to drink", comes at the end of the sentence, and its subject and object precede it. In an English translation of this sentence, the word order would be Subject–Verb–Object; in Japanese, the most natural word order is Subject–Object–Verb.[1] If we were to translate this sentence from Japanese to English without reordering the words, we might get "By the teacher tea was drunk". Such unnatural passivization is common in some statistical Japanese→English translation systems, and is a problem that our thesis aims to quash.

The second notable feature of Japanese grammar is that most words have explicit **case markers**. A word's **case** represents the function it plays in the sentence: subject, object, nounal modifier, etc. Japanese puts one syllable after most words to explicitly mark the word's case. In the above example, 'が' (the subject marker) marks '先生' ("teacher") as the subject of the sentence. Similarly 'を' (the object marker) marks 'お茶' ("tea") as the object of the sentence. These short case markers are sometimes called "particles" or "postpositions" because in Japanese they always immediately follow the word they attach to.

Because each word has its role in the sentence demarked in its surface form, the words in this sentence can be **scrambled** with the meaning and grammaticality of the sentence preserved, as long as the verb stays at the end.

(1.2) お茶を 先生が　　飲みました 。
tea-*Acc* Teacher-*Nom* drank .

[1] Approximately 75% of world's languages are Subject–Object–Verb [Crystal, 1997], so long-distance verbal reordering is a critical issue not only for Japanese→English translation.

"The teacher drank tea."

This scrambling also presents a challenge to existing translation systems and is addressed in our work.

The previous examples showed that in Japanese, the main verb always comes at the end of the sentence. Many patterns in Japanese are similar; verb phrases have the verb at the end, noun phrases have the noun at the end, and so on. To state this phenomenon with more formal linguistic terminology, the **head** of a phrase is the word in the phrase that determines its syntactic type; for example, the head of the English noun phrase "the girl who was sitting and drinking tea" is the noun "girl". Similarly, in Japanese:

(1.3) 正座して　お茶を 飲んでいた　女の子
sitting and tea-*Acc* was drinking girl
"the girl who was sitting and drinking tea"

Notice that the nounal head "girl" is at the end of the clause, while in English it is at the beginning of the clause. In general, we can say that Japanese is **head-final** while English is more **head-initial**. Swapping head orientation is a difficult aspect of Japanese→English translation. We next take a look at how this is handled in current translation systems.

1.2 State-of-the-art Japanese→English translation approaches

Current machine translations systems fall into three categories:

Rule-based sytems rely on hand-built syntactic parsers and many manually-edited transfer rules. Rule-based Japanese→English systems have been around for more than 30 years and are of high quality. Example: Yahoo! Japan Translation at `http://honyaku.yahoo.co.jp`.

Phrase-based systems are trained only on a large corpus of parallel text, from which they learn a set of multi-word phrases and a language model, without

using syntactic knowledge [Koehn *et al.*, 2003]. A well-known example is Google Translate at `http://translate.google.com`.

Hybrid systems combine a statistical (sometimes phrase-based) model with syntactic knowledge. One successful example is the dependency treelet system of Quirk *et al.* [2005].

In a nutshell, the translation system developed in this thesis classifies as a *hybrid* system. We started with the open-source **Moses** statistical phrase-based translator [Koehn *et al.*, 2007], and modified it to incorporate a syntactic parse analysis to improve reordering decisions.

The largest available collection of Japanese–English parallel text is Utiyama's Patent Parallel Corpus [Utiyama *et al.*, 2007], so in this thesis we focus on examples from the domain of patent translation. Gloss 1.4 shows an example from our test corpus. Translations from the best available Japanese–English translation systems are given in Table 1.2.

(1.4) プリアンプ 3は　　入力された　再生信号を　　増幅して　ＡＧＣアンプ4へ 出力する 。
Preamp　3-*TOP* input-*Passive* reproduction signal-*Acc* amplify and AGC amp 4-to　output　.
"The preamp 3 amplifies an input reproduction signal, and sends out to an AGC amplifier 4."

Table 1.1: Comparison of translations of Gloss 1.4.

MosesImproved	MosesBaseline	Google	Yahoo
The preamplifier 3 amplifies the reproduced signal, which is output to the AGC amplifier 4.	The preamplifier 3, the input playback signal is amplified and output to the AGC amplifier 4.	3 preamp input signal is amplified by playing the AGC amplifier, the output 4.	Pre-amp 3 amplifies an input reproduction signal and outputs it to AGC amplifier 4.

Translations from Yahoo and MosesImproved are very good, while offerings from MosesBaseline and Google are unnatural or incorrect. In the next section, we will discuss the shortcomings of these systems.

1.3 State-of-the-art foibles

In Table 1.2, YAHOO is the translation from Yahoo Japan Translation, a rule-based system under development since 1987 [Cross Language, 2008].[2] The other three systems are phrase-based. Systems GOOGLE and MOSESBASELINE translate content words and idioms accurately, but for the most part eschew syntactic analysis. A lack of syntactic sensibility leads to several systematic errors.

Most noticeably, word order is incorrect. In both the GOOGLE and MOSESBASELINE translations, the verb "amplified" follows its object "reproduction signal". Phrase-based systems employ several scoring functions for ranking hypothesis translations. One such **feature function** penalizes each reordered phrase. This feature function is helpful for translating from, for example, French→English, where word order is largely preserved, but is not useful for Japanese→English translation, and encourages verbs to stay at the end of their clause. The Google translation blithely leaves "output" as the last word in the sentence.

The component most responsible for reordering in phrase-based systems is the **language model**, which gives a higher score to translations composed of n-grams that appeared often in a large corpus of English training text. The language model helps to encourage phrases to reorder into a grammatical a translation, but the grammatical word order chosen by the language model often does not maintain the meaning of the original Japanese sentence. For example, consider the Moses baseline: "The preamplifier 3, the input playback signal is amplified..." The main verb is made passive, keeping the original verb-object Japanese word order while remaining grammatical. In the process, the subject (what is performing the amplification) is separated from the verb. The meaning of the sentence is lost.

In contrast, rule-based systems, like Yahoo's, perform well translating patent data, for two reasons. First, highly regular legalese can be parsed by handwrit grammars, which are composed of thousands of special-case rules that occasionally break down

[2]Yahoo Japan Translation uses software from Cross Language, a company specializing in Japanese-Chinese-Korean translation services. Its translation quality is much better than Systran, another eponymous system translation services company's product, which is what Yahoo Babel Fish uses for its backend [Yahoo, 2008].

on more colloquial text. More colloquial text, in contrast, omits many case markers and understood pronouns so is much harder to parse. Second, literal translations are acceptable because idiomatic patterns are rare in patent text. Rule-based systems must have special rules for any expression which it translates idiomatically.

In this thesis, we improve the phrase-based decoder Moses to perform syntactically-motivated reordering, and thus aim to achieve the best of both worlds. The translation from MosesImproved uses a syntactic dependency analysis to improve on the Moses baseline. This method is introduced nextly.

1.4 Dependency analysis in a phrase-based translator

The major contribution of this thesis is a method to integrate syntactic dependency information into the Moses phrase-based translator. The idea is to translate a dependency tree, instead of a flat sentence. For example, the dependency parse identifies a sentence's main verb and object. During translation, we can give higher scores to translation hypotheses that put the main verb before its object.

Let's look at how this works for Gloss 1.4. Figure 1-1 shows its dependency parse. Arrows indicate dependencies. For instance, the arrow between them indicates that

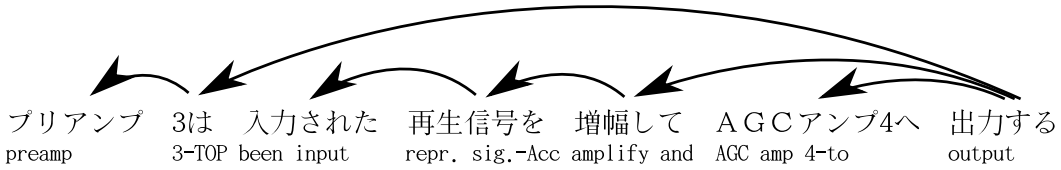

Figure 1-1: Preamp dependency parse example.

"amplify" depends on "reproduction signal-*Acc*". Further observing that "reproduction signal-*Acc*" has accusative case, and knowing that the target language English has Subject–Verb–Object order, the translator can prefer to translate the verb "amplify" before it translates its object "reproduction-signal". We will codify this preference by introducing a feature function in Moses that counts occurrences of a verb being translated before its object.

In addition, we will introduce feature functions for a range of grammatical constructs: a feature that counts when relative clauses are translated after the noun they modify, one that counts when genitive modifiers are translated after the noun they modify, and so on. We could have a feature for every part-of-speech and case pairwise combination. Furthermore, we introduce a cohesion constraint in the same vein as [Cherry, 2008]

We discriminatively train the weights of these features to identify the most useful features and maximize translation quality. This discriminative training step is important to tune the system for the grammatical features of the target language. While the verb-before-its-object feature function identifies good English translations, if we were translating into Japanese, we would give a negative weight to the verb-before-its-object feature. This setup would correctly prefer to translate Japanese verbs after their objects.

1.5 Syntactic reordering in the preprocessor

We experimented with one more technique to reorder the Japanese training data into an English-like word order before running Moses training (following [Wang *et al.*, 2007]). When translating an unseen Japanese sentence, we first **preorder** it into this English-like word order, then translate preordered Japanese sentence with the specially-trained Moses setup. With this approach, the burden of reordering phrases is pushed to a syntactic preprocessing step, and the Moses translator itself can perform a largely **monotonic** (no reordering) translation, at which it excels.

The challenge is to build an algorithm that reorders a Japanese sentence into a pseudo-Japanese sentence that has the same words but in English-like word order. In this thesis I describe two such algorithms. The first is fast and naive, and simply reverses the order of all tokens after splitting the sentence at punctuation and 'は', the topic marker. The second algorithm uses three linguistically-motivated heuristics for flattening a tree formed from a dependency parse.

For illustration, Gloss 1.5 shows the preamp sentence reordered with the naive

reverse preprocessor, which will be described in detail in Section 4.2.

(1.5) は 3　プリアンプ する 出力 へ 4 アンプ ＧＣ Ａ て し 増幅 を 信号 再生　たれ さ 入力 。
TOP-3 preamp output to 4 amp GCA and amplify *Acc*-repr. signal input-*Passive* .

To complete the example, we could insert several function words into the English gloss given above to complete a fluent sentence: "The 3 preamp outputs to 4 amp ACG and amplifies the reproduction signal that has been input." This shows that we could translate the preprocessor-reordered Japanese sentence into English with a monotonic translation.

In our experiments, we found an improvement in translation quality using the naive reverse preprocessor. Surprisingly, we saw a *smaller* improvement using the linguistically-motived smarter preprocessor, which usually produced more accurately English-like pseudo-Japanese.

We achieved the best translation quality when combining approaches: use the reverse preprocessor and an assortment of dependency-motivated feature functions at optimal weights. Altogether, we achieved a BLEU score improvement of 27.96→28.74

Chapter 2

Related Work

This chapter outlines recent work on statistical reordering models in machine translation. Methods span a wide gamut: preprocessing techniques, reranking techniques, linguistically-informed reordering constraints, local distortion models, tree-to-tree and tree-to-string translators, and dependency treelet systems.

2.1 Reordering during preprocessing

Collins *et al.* [2005] introduced a very effective technique for building a phrase-based system with long-distance reordering ability. Working on German→English, they wrote rules to transform a deep parse of the German sentence so that its words read in English word order. They parse the German training data, apply these rules to transform it into English word order in a preprocessing step, then train a phrase-based system on the reordered data. Before translation, they perform the same reordering on the input sentence. This led to a significant improvement in English output word order. Wang *et al.* [2007] followed up with analogous experiments for Chinese→English.

In Chapter 4, we apply the same technique to Japanese→English translation, with two twists. First, we introduce a trivially computable reordering algorithm for putting Japanese into English word order, in addition to a reordering algorithm that flattens a Japanese dependency tree into English word order. Second, our algorithms keep

the dependency information from the tree imbedded in the reordered sentences so that the dependency analysis can be used by the decoder to make smart reordering decisions at decoding time.

Kanthak *et al.* [2005] further developed the preordering technique. Their system automatically learns how to reorder source sentences into target language word order from monotonization of training data word alignments. However the weakness of their baseline decoder, which failed to translate 37% of their Japanese test corpus, makes it difficult to tell how effective their automatically-trained source-side reorderer is.

Li *et al.* [2007] takes the idea of Kanthak *et al.* one step further. First they trained a statistical source-side reordering model, which predicts whether a node of a tree should keep its children in order or invert them, by using word alignments and deep parses of the source sentences of the training data. To translate a sentence, they generate the 10 best preorders with their reordering model, then translates all of the preorders with a phrase-based decoder (using a maximum distortion limit of 4) and out of the 10 pick the translation with highest combined source-side reordering model score and decoder score. They worked with Chinese→English and achieved an improvement over their no-preordering baseline of the same magnitude as Wang *et al.* [2007]. The advantage of Li *et al.*'s work is that there is no need for handwrit tree reordering rules.

2.2 Reranking phrase-based system output

Och *et al.* [2004] tested a range of global syntactic features on 1000-best output of a phrase-based system. They found no significant improvements from statistical features, including target-side parse tree probability, tree-to-string model probability, tree-to-tree model probability, and word alignment scores from a Tree Adjoining Grammar. One interesting finding was that a state-of-the art statistical parser tended to assign higher probability to ungrammatical machine translation output than to human-translated references. This is one reason that we chose to incorporate only a source-side dependency analysis.

Nichols *et al.* [2007] developed a Japanese→English Moses system and a separate rule-based translator based on three man months of handcrafted transfer rules. The parser, also based on handwrit rules, can parse 65% of sentences, and the transfer rules succeed 33% of the time. When available, their system picks the rule-based translation (about 13% of the time) and otherwise falls back on the Moses translation. They found that the rule-based system makes poor word choices, while the Moses system has trouble preserving the structure of the sentence.

2.3 Reordering models for phrase-based systems

Zens *et al.* [2004] implemented several reordering constraints in a phrase-based Japanese→English decoder.[1] The first constraint is the same as the maximum distortion limit in Moses (see Section 3.5.1) and the second is the "ITG constraint", where only reorderings that could have been made by either straight or inverted combo of contiguous "blocks" are allowed. Each block is a combination of phrase pairs continguous on both the source and target side. These constraints aid in ruling out certain reorderings that are more probable to be bad, but do not aid in identifying reorderings that preserve meaning of the original sentence.

Kanthak *et al.* [2005] subsequently built a decoder that takes as input a weighted finite-state reordering automaton with constraints based on the work of Zens *et al.* [2004]. They added additional reordering constraints under which words at the end of a sentence are translated first, in a special case for Japanese. Otherwise transducer paths are weighted to prefer monotonic translation. It is hard to tell how well their reordering automaton works, because they compare it to a baseline that allows no reordering.

Tillmann [2004] introduced a local, lexicalized, phrase orientation model. This model, now implemented in Moses and described in detail in Section 3.4.5, predicts whether a phrase swaps position with the previous or next phrase based on phrase

[1] The "decoder" is the program that searches for the best translation of a sentence; we examine its anatomy in Section 3.2.

alignment of the training data. In a later work, Tillmann and Zhang [2005] built a maximum-likelihood trained log-linear model to predict the same thing. Al-Onaizan and Papineni [2006] developed a similar model that assigns a probability distribution over possible relative jumps conditioned on source words. In another alike technique, Kuhn *et al.* [2006] wrote a decoder that chooses the next phrase to translate based on a lexicalized decision tree trained on phrase alignment of the training data. As part of their discriminatively trained system with millions of features, Liang *et al.* [2006] added thousands of phrase-orientation features for each part of speech pair, but it is difficult to gauge their utility because their decoder allowed very limited reordering.

Xiong *et al.* [2006] developed a similar reordering model that estimates the probability of two given "blocks" combining in straight or inverted order, where a block is a pair of source and target contiguous sequences of words. (A block could be one phrase pair, or a combination of multiple contiguous phrase pairs.) They employ the first and last word of each block as features, and use phrase alignments from the training data as reordering examples in a maximum-entropy framework. Zhang *et al.* [2007] improved on Xiong *et al.*'s model by incorporating part of speech and dependency features conditioned on block boundary words. It is unclear how well these block reordering models can handle long-distance reordering with a series of independent decisions based only on block boundary features. After translating a Japanese sentence, the best-scoring translation may never have compared the position of the main verb relative to its object.

Cherry [2008] incorporated dependency information into Moses and added a feature function that counts how often a dependency subtree's translation is interrupted by translating a different part of the tree. Cherry found that sentences translated cohesively tend to receive higher BLEU score and human judgment than uncohesive translations. In Section 3.6.2, we describe a comparable cohesion feature that we incorporated in our experiments.

2.4 Hierarchical phrases

Chiang [2007] introduced a model akin to a phrase-based system but with hierarchical phrases. Each phrase can include nonterminals where other phrases can nest. Long-distance reordering patterns can be learned automatically with this mechanism. For example, the Chinese→English phrase pair ≪[1] 的 [2], the [2] of [1]≫ swaps the position of its two arguments, which could be arbitrarily long.[2] The major idea is that the hierarchical phrase model is *formally* syntax-based in that it uses the Synchronous Context-Free Grammar formalism, but not linguistically syntax-based, because it induces a grammar from a parallel text without relying on any linguistic assumptions or annotations (like the Penn Treebank). Because to our knowledge Chiang's model has only been applied to Chinese–English translation, it is unknown how well hierarchical phrases can do as the only motivators of long-distance reordering in a language pair like Japanese→English that requies a lot of it.

2.5 Tree-to-string translation

Systems that decode by translating a parse tree bottom-up have recently come into vogue. The decoder of Riezler and Maxwell III [2006] feeds dependency parse snippets into a grammar generation component and scores with feature functions similar to a phrase-based decoder, using dependency snippet transfer rules instead of phrase pairs. Huang *et al.* [2006] offer a similar setup using parse-tree-to-string transducers, and Liu *et al.* [2006] contribute a system using tree-to-string alignment templates.

In general, tree-based decoders must tackle difficult challenges in efficiency and how to integrate varied information sources like a language model. This thesis avoids such issues by incorporating a source-side dependency analysis in an existing phrase-based decoder, which translates in an efficient left–to–right manner with an easily extendable log-linear scoring model. Still, by decoding in a flat string-to-string manner, we make at least theoretical concessions in preserving sentence meaning and

[2] In a Japanese→English, we might see a very similar phrase pair ≪[1] の [2], the [1] that [2]≫. We introduce a feature to handle this inversion in Section 5.6.2.

target-language grammaticality.

2.6 Tree-to-tree translation

Ding and Palmer [2005] focus on dependency-tree to dependency-tree translation using a synchronous dependency insertion grammar induced from the training data, but do not build a head-reordering model for flattening the resulting dependency tree, so they systematically generate translations such as "foreign financial institutions the president of". Correspondingly, their system could not model the head-initial to head-final inversion crucial for Japanese→English translation. Lin [2004] developed a similar tree-to-tree system based on assembling linear paths through a source-side dependency tree, but like Ding and Palmer they incorporated no language model or discriminative reordering model, which led to disappointing BLEU scores.

Cowan *et al.* [2006] stepped it up with a system based on Aligned Extended Projections, which consist of a pair of corresponding clausal tree structures extracted from the training data using deep parsers for both source and target languages. This system excels at clausal translation, but does not yet model how to reorder clauses. Clausal reordering is not critical for their language pair, German→English, but is important when translating Japanese sentences, which often have deeply nested dependencies ordered oppositely compared to English.

2.7 Dependency treelet translation

In their "dependency treelet" system, Quirk *et al.* [2005] parse the source side of the training data, project these dependency trees onto the target side using word alignments, then extract dependency treelet pairs. A treelet is defined to be an arbitrary connected subgraph of the dependency tree. The decoder covers the source dependency tree with treelet pairs bottom-up and scores hypotheses with a log-linear model incorporating typical features, such as language model and word alignment probabilities, and a novel order model.

Their order model assigns a probability to the word order of a target tree given a source tree. This order model makes the assumption that the position of each child can be modeled independently in terms of its position relative to its head (parent in the dependency graph). Their features model whether a modifier is ordered to the left or right of its head, and how far away, with features parameterized on word and part of speech of the head and modifier. Quirk *et al.* [2005] train the order model as a decision tree. Menezes *et al.* [2006] later upgraded it to a log-linear model with features chosen to maximize performance on a development set. Chang and Toutanova [2007] introduced a global order model that ranks n-best dependency tree output of the treelet system using local features that capture head-relative movement and global features that capture the surface movement of words in a sentence.

In Section 3.6.1, we introduce a similar set of features that model head-relative movement using a source-side dependency parse. We additionally condition our features on the case of the modifier and head, and simplify the model so it predicts only if a modifier should be on the left or right side of the head. We eschew lexicalized features in our model, but they could easily be added by further parameterizing our features, which we leave as future work.

Menezes and Quirk [2007] improved on their initial treelet approach with the "dependency order template" system that avoids the combinatorial explosion of reordering treelets that they encountered in their 2005 effort, which necessitated strict pruning of the search space. They introduce order templates, which are unlexicalized transduction rules mapping dependency trees containing only parts of speech to unlexicalized target language trees. These order templates are extracted from source-side dependency trees and word alignments of the training data.

At translation time, order templates are combined with relevant treelet translation pairs to construct lexicalized transduction rules. Menezes and Quirk cite two advantages of this approach: the decoder needs only to consider reorderings that are captured in some order template, and reordering knowledge can generalize to uncommon words because the order templates specify only part of speech. Our feature functions play a role similar to order templates and share the positives: they aim the

decoder's beam so that correct reorderings are not pruned, and they can pick up the fully-lexicalized phrase table and language model because our features condition only on part of speech and case.

Xiong *et al.* [2007] also incrementally improved the treelet system of Quirk *et al.* [2005] to support discontinuous output phrases and generalized treelets with imbedded variables, in a manner reminiscient of Chiang's hierarchical phrases.

Chapter 3

Syntactic Feature Functions for Phrasal Reordering

This chapter introduces the major contribution of this thesis: syntactic feature functions for a state-of-the art phrase-based machine Japanese→English translation system that significantly improve reordering decisions. These features score English translation hypotheses using a dependency parse of the source Japanese sentence. We add many such feature functions, one per dependency relationship we wish to model, and discriminatively train their weights. The most useful single feature increased BLEU score 27.97→28.35. Combining the three most useful additional features, we achieved a 27.96→28.54 BLEU increase.

We give a whirlwind picture of the Moses phrase-based translation system and show where these new feature functions fit in the translation process. Then we detail the feature functions themselves.

3.1 Moses phrase-based translation system

Phrase-based systems represent the state of the art in machine translation; phrase-based systems, like Google's, have dominated the NIST Machine Translation Evaluation, held yearly since 2001 [NIST, 2006].

Moses is another high-quality phrase-based translation system [Koehn *et al.*, 2007].

Moses is free software and actively developed by many researchers around the world (most notably at the University of Edinburgh) and has been used as a baseline system for several major translation workshops [WMT Baseline, 2007; Fujii *et al.*, 2007]. It includes open-source implementations of everything needed to build a translation system between any language pair. The most important of these components is the **decoder**, which performs the actual search for the best Japanese translation of an input English sentence.

3.2 Framing the search problem

The job of the decoder is to search for the best English translation e (of length I) of a given Japanese sentence f (of length J). The decoder chooses the English sentence with highest probability:

$$\hat{e} = \text{argmax}_{e_1^I} \left\{ Pr\left(e_1^I | f_1^J\right) \right\} \tag{3.1}$$

Because the search space is all possible English sentences, formulating this search problem efficiently is challenging, and will be an important topic of discussion later.

We can use Bayes' rule to rewrite the probability in Equation 3.1 as:

$$\hat{e} = \text{argmax}_{e_1^I} \left\{ Pr\left(e_1^I\right) \cdot Pr\left(f_1^J | e_1^I\right) \right\} \tag{3.2}$$

Framing the search problem in this way, called the source-channel approach [Brown *et al.*, 1993], is appealing in theory, if we have access to the true probability distributions $Pr\left(e_1^I\right)$ and $Pr\left(f_1^J | e_1^I\right)$. In practice, to model $Pr\left(e_1^I\right)$, we use an n-gram language model, and to model $Pr\left(f_1^J | e_1^I\right)$, we use phrase co-occurrence probabilities learned during training. These methods provide poor approximations to the true distributions, so the combination in Equation 3.2 may be suboptimal. One more problem with this approach is that it is not clear how to extend this system with more dependencies, like additional data or scoring functions.

To combat these problems, Och and Ney [2001] introduced a maximum entropy

model that directly models the posterior probability, and leads to the following decision rule:

$$\hat{e} = \text{argmax}_{e_1^I} \left\{ Pr\left(e_1^I \middle| f_1^J\right) \right\} \tag{3.3}$$

$$= \text{argmax}_{e_1^I} \sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J) \tag{3.4}$$

In Equation 3.4, we have a set of M feature functions $h_m(e_1^I, f_1^J)$, $m = 1, ..., M$ used to score translation hypotheses. For each feature function, there is a model parameter λ_m that gives a relative weight to the feature.

Now we have framed the translation problem in terms of writing feature functions that can identify good translations and setting the weights for those features. We can recover the source-channel approach as a special case if we make our first feature function the log of the language model probability, and our second an estimate of $Pr\left(f_1^J \middle| e_1^I\right)$ based on hidden phrasal alignments (covered in the next section) and co-occurrence counts of aligned phrases in the training data. We can furthermore add as many feature functions as we wish, if we think they may be aidant in distinguishing good translations from bad. Before we introduce more feature functions in Section 3.4, we must develop the phrase-based translation model that Moses is built on.

3.3 Phrase-based translation

To translate an input Japanese sentence, we will segment it into phrases, translate each phrase into English, then reorder those phrases to produce the output English translation. In the context of this discussion, a **phrase** means simply a contiguous sequence of words in either language; it is not used in any linguistic sense. The building block of phrase-based systems is the **phrase pair**, which comprizes a Japanese phrase and its English translation. Training our translation model consists of automatically learning a **phrase table** from the parallel training corpus. For more detail on this training process, see [Koehn, 2007].

To codify the notion of phrases into our translation model, we follow [Och and

Ney, 2004] and introduce a hidden phrasal decomposition by segmenting the Japanese sentence f_1^J and English sentence e_1^I each into a sequence of K phrases ($k = 1, \ldots, K$):

$$f_1^J = \tilde{f}_1^K, \tilde{f}_k = f_{j_{k-1}+1}, \ldots, f_{j_k} \tag{3.5}$$

$$e_1^J = \tilde{e}_1^K, \tilde{e}_k = e_{i_{k-1}+1}, \ldots, e_{i_k} \tag{3.6}$$

We further introduce a hidden phrasal alignment π_1^K between the Japanese phrases $\tilde{f}_1^K$ and the English phrases $\tilde{e}_1^K$. This alignment is a permutation of the English phrase positions $1, \ldots, K$, so that $\tilde{e}_k$ and $\tilde{f}_{\pi_k}$ are translations of each other. We finally define z_k as the phrase pair (a pair of strings) that is used to translate the kth Japanese phrase:

$$\tilde{e}_k \overset{z_k}{\longleftrightarrow} \tilde{f}_{\pi_k} \tag{3.7}$$

Hence, in the decoding process, we simultaneously search for 1) the optimal segmentation of the Japanese sentence into phrases; 2) the optimal English translation for each phrase; and 3) the optimal way to order these phrases into an English sentence. We use hidden variables z_1^k, a vector of the phrase pairs used, and π_1^K, their permutation from Japanese to English order, to help us score hypotheses. With this model, our feature functions take the functional form

$$h(e_1^I, f_1^J, \pi_1^K, z_1^K). \tag{3.8}$$

3.4 Baseline feature functions

Our baseline Moses setup uses the following feature functions.

3.4.1 Language model

In our experiments, we used a 5-gram language model:

$$h_{\mathrm{LM}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = \log \prod_{i=1}^{I+1} p\left(e_i \middle| e_{i-4}, \ldots, e_{i-1}\right) \tag{3.9}$$

The language model picks translations which look like grammatical English, without regard to whether or not they are an adequate translation of the original Japanese sentence.

Al-Onaizan and Papineni [2006] illustrated the inability of the language model to discriminate correct reorderings by itself. They rearranged a corpus of English sentences into Arabic word order, then tried to translate them into English with a phrase-based decoder and no distortion model except an English language model. As they increased the maximum reordering limit so that words could freely reorder, English word order recovery rapidly deteriorated. While the language model is very important in producing grammatical English, we must rely on complementary reordering models to preserve the meaning of the original sentence.

3.4.2 Translation model

Moses models $Pr\left(e_1^I | f_1^J\right)$ by scoring each phrase separately:

$$h_{\mathrm{TM}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = \log \prod_{k=1}^{K} p\left(z_k | f_{j_{\pi_k-1}+1}^{j_{\pi_k}}\right) \tag{3.10}$$

In Equation 3.10, the phrase translation probability distribution is estimated by relative unsmoothed frequency in the training data. Moses also includes inverted probabilities to model $Pr\left(e_1^I | f_1^J\right)$, which are otherwise analogous to the above, and the "lexical weighting" of each phrase, which is described on p. 5 of [Koehn *et al.*, 2003].

These translation model features together pick translations which have all of the right content words, but not necessarily in the right place (target language phrases may be in the wrong order) or with agreeable dependencies (two phrase translations might make sense independently, but be laughable together because of word sense ambiguity).

3.4.3 Word and phrase penalties

These are simple features to count how many words long the hypothesis is:

$$h_{\textsc{WordPenalty}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = I \tag{3.11}$$

And how many phrases long it is:

$$h_{\textsc{PhrasePenalty}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = K \tag{3.12}$$

These features provide a straightforward method to tune output translation length. One reason this is important is that our Japanese preprocessor splits sentences into many more tokens than there are English words in an optimal translation.

3.4.4 Distortion penalty

This feature is roughly a measure of how far phrases have been reordered compared to a monotonic translation. This is computed by the negative sum over the distance (in the source language) of phrases that are consecutive in the target language:

$$h_{\textsc{Distortion}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = -\sum_{k=1}^{K+1} \left| j_{\pi_k - 1} - j_{\pi_{k-1}} \right| \tag{3.13}$$

where j_{π_0} is defined to equal 0 and $j_{\pi_{K+1}-1}$ is defined to equal J.

In a Japanese→English system, this feature is of little help to distinguish quality translations. Because of the vast difference in English and Japanese word order, non-monotonic translation is the norm rather than the exception. For most of our systems, this feature got a very low or negative weight ($\lambda_{\textsc{Distortion}}$) after parameter tuning. With a negative weight, this feature *encourages* non-monotonic translations.

If we consider language pairs with similar word order like French→English, for which the first phrase-based translation systems were developed, this distortion penalty is extremely beneficial. In the words of Och and Ney [2004], it "simply takes into account that very often a monotone alignment is a correct alignment."

3.4.5 Local lexical reordering

Finally, Moses includes a set of features that improve local reordering decisions. They model how often a phrase is translated monotonically relative to the phrase before it, how often a phrase swaps place with the phrase before it, and how often a phrase is translated discontinuously relative to the phrase before it. Additionally, analogous features are included for modeling how a phrase is ordered relative to the phrase after it.

Knowing whether a phrase prefers monotonic translation or to swap with a neighbor is very useful for a language like Spanish, where these local lexical reordering features give a significant gain in BLEU score, as shown in Appendix C of [Koehn *et al.*, 2007]. For example, in Spanish, adjectives follow the noun they modify; "green salsa" is, deliciously, 'salsa verde' [Knight, 1999]. In a Spanish→English translation system, Moses's local lexical reordering features can give a higher score to translation hypotheses that correctly swap adjectives to come before the noun they modify.

However, these features are not sufficient for Japanese→English translation. First, a local reordering model offers little help to reorder verbs from the end of a Japanese sentence. Often the Japanese verb must leapfrog many phrases to get to its English proper spot between its subject and object. In this case, the Moses local reordering features can only tell the decoder to reorder the verb, not to where.

Second, the scrambling property of Japanese means that contiguous phrases do not necessarily have a relation to each other. Therefore statistics counting how a phrase is ordered relative to the previous and next phrase are not very meaningful. Furthermore, these counts are learned from phrase alignments induced from the training corpus. Based on our personal observations of our Japanese–English data, phrase alignments in the training data are very noisy and not reliable for learning reordering patterns.

Still, these local reordering feature functions have some utility for Japanese→English translation. One merit is that they can handle the agglutinative morphology of Japanese verbs. To inflect a Japanese verb (to make it negative, past tense, polite, or otherwise) one appends morphemes to the end of the verb. This is demonstrated by

segmented output from our Japanese preprocessor in Gloss 3.1. In this example, the verb "gaze" has an ending that makes it negative together with a politeness marker, and an ending that puts it into past tense together with another politeness marker.

(3.1) 私 は　星　を 眺め　ませ　ん でし　た　　。

I　*TOP* stars *Acc* gaze at [polite] not [polite] [past tense] .

"I did not gaze at the stars."

It is natural to translate this sentence with phrase pairs: ≪私 は, I≫ ≪星 を, the stars≫ ≪眺め, gaze at≫ ≪ませ ん でし た, did not≫. The local reordering feature should identify that the inflection "did not" should swap with the verb stem "gaze at", and that this verb stem should in turn swap with its object "stars". These two reorderings lead to the correct permutation of the English phrases.

Notice that one word is often split over more than one phrase during translation because of abundant Japanese morphology. Here, the verb '眺めませんでした' ("did not gaze at") is translated as part of two distinct phrases: '星を眺め' and 'ませんでした'.

3.5 Beam search

It is clear that Moses needs long-distance reordering features to effectively translate Japanese to English, but we must be careful that our features are efficiently computable during the decoding process. This section introduces the decoding machinery and the constraints it imposes on the structure of our features.

Moses, like most phrase-based decoders, performs the search in Equation 3.4 with an iterative **beam search** [Koehn *et al.*, 2007]. It is called a beam search because the decoder explores the space of possible translations breadth-first, translating one phrase at a time, but quickly discards very low-scoring translations. This leads to the possibility of search errors, where the highest-scoring translation under our model is not found. Practically, these errors are not a prohibitive problem; still, it is useful if features can identify promising translations as early on as possible to prevent them from being discarded.

The decoder keeps a stack of hypotheses. It **expands** each hypothesis in the stack by translating one **uncovered** Japanese phrase, appending this translation to the end of its work-in-progress English translation, and adding the new resulting hypothesis to the stack. The Japanese phrase that was translated becomes **covered** in this new hypothesis. In this way the decoder assembles English hypothesis translations from left to right, translating one Japanese phrase at a time. The order in which it picks Japanese phrases to translate determines the word order of the English output sentence.

These are the critical data that each hypothesis contains:

- phrase pair translated by this hypothesis
- back link to the previous hypothesis that this one expands, which allows us to recover the English translation
- bit vector representing which Japanese words have been translated
- vector of feature function scores
- score, computed by taking the dot product of the feature scores vector with the feature weight vector

Let us take a look at examples of translation hypotheses for the preamp example introduced in Chapter 1, reproduced here with spaces between words.

(3.2) プリアンプ 3 は　入力 さ れ た 再生 信号を　増幅 し て　Ａ ＧＣ アンプ 4 へ 出力 する 。
Preamp 3-*TOP* input-*Passive* repr. signal-*Acc* amplify and AGC amp 4-to output .
"The preamp 3 amplifies an input reproduction signal, and sends out to an AGC amplifier 4."

After translating several phrases of the preamp example, this is a promising hypothesis:

Preamp Hypothesis #149442	
Expands	#84738
Covers	プリアンプ 3 は . . . . . ≪を 増幅 し≫ . . . . .
Phrase pair	≪を 増幅 し, amplifies the≫
Features	$<$ Distortion $= -12$, WordPenalty $= 5$, LM $= -25.821, \ldots >$
Score	$-101.988 +$ future cost $- 103.270 = -205.258$

(Ellipses represent uncovered words in the coverage vector.)

Hypothesis #149442 is expanded into many hypotheses, and among the best is Hypothesis #318530:

Preamp Hypothesis #318530	
Expands	#149442
Covers	プリアンプ 3 は ≪入力 さ れ た 再生 信号≫ を 増幅 し . . . . .
Phrase pair	≪入力 さ れ た 再生 信号, reproduced signal , which is≫
Features	$<$ Distortion $= -21$, WordPenalty $= 10$, LM $= -40.422, \ldots >$
Score	$-102.947 +$ future cost $- 102.049 = -204.995$

To complete this example, Table 3.1 lists a very small subset of the phrase table that could next be used to expand Hypothesis #149442.

3.5.1 Search efficiency

Because the search space of all possible English sentences is so huge, the Moses decoder takes a number of measures to keep the search process efficient. The first is a hard limit on how far phrases can reorder. With a maximum distortion limit *MaxDistortion*, the next phrase the decoder picks to translate must start within *MaxDistortion* words from the leftmost uncovered word. For many language pairs, this limit is important for high quality translations, and setting $MaxDistortion > 6$ begins to hurt translation quality [Koehn *et al.*, 2005]. For Japanese→English we

て	with the
て	through
て	based on
ａ ｇ ｃ アンプ	the agc amplifier
ａ ｇ ｃ アンプ	agc (automatic gain control) amplifier
ａ ｇ ｃ アンプ	including the agc amplifier
へ	into
へ	on
へ	to a
へ 出力	is output to the
へ 出力	output
へ 出力	to output
へ 出力 する	outputs to the
へ 出力 する	to output the
へ 出力 する	and outputs the result to the
する	with
する	, the
する	be

Table 3.1: Phrase table excerpt.

found that having no distortion limit gave highest translation quality. However, decoding our test set with no distortion limit (defined as $MaxDistortion = 0$) takes on average 37 seconds per sentence, which is 5 times longer than with $MaxDistortion = 9$. Thus there is an important tradeoff between quality and speed. (See Section 5.4 for distortion limit experiments.)

Each hypothesis is scored before it is added to the stack, and at each step, the decoder prunes the stack to keep only the highest-scoring hypotheses. There is another quality-speed tradeoff in setting the maximum size of the hypothesis stack. The default stack size is 100; increasing this to 200 improves quality slightly (28.46→28.63 BLEU) but also causes translation to take almost twice as long. One more optimization Moses implements is to recombine identical hypotheses (as measured by which Japanese words have been translated and end of the English translation), and keep only the higher-scoring hypothesis. We now must formulize how to score a hypothesis which may have a set of uncovered Japanese words yet to be translated.

We first decompose each feature function into a sum of the contributions from each English phrase used in the translation. This allows us to calculate a feature's value

for a hypothesis by adding together 1) the contribution of the last translated English phrase and 2) the previous value of the feature in the hypothesis this one was expanded from. As a simple example, consider the word penalty feature, WordPenalty, of Equation 3.11, which equals the number of words in the sentence. Let's say that a hypothesis A is expanded into hypothesis B by adding English phrase C of length C.length to the end. The value of B's WordPenalty feature is equal to the value of A's WordPenalty feature plus the contribution of C, which is C.length.

A hypothesis's score is then the dot product of the feature scores vector with the feature weight vector. In addition, because the pruning compares translations that may have translated differing subsets of Japanese words, we also add a heuristic to the score that estimates the future cost of translating the uncovered Japanese words.[1]

Unfortunately, many useful feature functions do not decompose nicely into contributions from each used phrase pair, and we are unable to incorporate them into our beam search. One approach to incorporate such global features, used for example by Och *et al.* [2004], is to use them in an n-best reranking step. With this method, the efficacy of the features is limited by the quality of the translations in the n-best list; if the n-best list does not contain translations with the needed long-distance reordering, there is no hope for the reranker to pick a good translation. As a result, Och *et al.* could not achieve a significant improvement in Chinese→English translation quality with their global syntactic feature functions reranking a 1000-best list. Hence we will focus our attention on designing only features that can be integrated directly into a beam search.

Let us recap the constraints that the decoder imposes on the space of possible feature functions:

- Must decompose into a sum of contributions from each phrase pair used translation.

- Each such contribution must be a function of only

[1] See [Och and Ney, 2004] for the derivation of such a heuristic, and [Koehn, 2007] for an explanation of how it is implemented in Moses.

- The input Japanese sentence.
- Which Japanese phrase was last translated to expand this hypothesis, and the English phrase used for their translation.
- A bit vector representing which Japanese words have been translated.

3.6 Long-distance reordering feature functions

With the previous discussion in mind, we aim to build long-distance reordering features with multipronged merits:

1. Model reordering over an arbitrarily long distance
2. Consistently perform head-final to head-initial reordering
3. Effective even in the wake of Japanese scrambling
4. Resistant to noisy word alignments in training data
5. Applicable to any language pair
6. Computable efficiently in a phrase-based decoder

Our features will use a dependency parse and count the number of times a certain dependency pattern occurs. One example is a feature that counts how many times in a sentence a verb is translated before its object. If we give this feature high positive weight, it will cause the decoder to prefer sentences with verbs preceeding their objects, as is correct English. If instead we give this feature negative weight, the decoder will prefer sentences with verbs coming after their objects, as would be preferred for translating into other languages, like Korean, Hindi, or another Subject–Object–Verb language.

Discriminative training can automatically assign optimal weights to optimize translation quality on a development corpus [Och and Ney, 2001]. In this way, our translation system does not need to know that English is a Subject–Verb–Object language, or any grammatical property of English; these properties are learned automatically

during discriminative training. To emphasize the applicability of these features to any language pair, we use **source** and **target** language to refer to the languages we are translating to and from.

We also introduce two more types of features in addition to these pairwise dependency pattern counters. One encourages cohesively translating all words of certain linguistic phrases before moving on to another phrase, and one discourages reordering phrases across punctuation marks.

We now introduce notation that will allow us to formally define these features. We view each hypothesis as a state transition, wherein one new phrase is translated. As it translates left–to–right one phrase at a time, the decoder assembles a sequence of state transitions. When all source phrases have been translated, the decoder's sequence of state transitions maps to $(e_1^I, f_1^J, \pi_1^K, z_1^K)$, which feature functions score. In line with the discussion in Section 3.5.1, feature functions that can be efficiency implemented in the decoder must decompose into contributions from each state transition.

We use the variable q to denote a hypothesis. We define q_k as the kth hypothesis in the decoder's state transition sequence underlying $(e_1^I, f_1^J, \pi_1^K, z_1^K)$. Then we can define efficiently-computable feature functions in the form

$$h(e_1^I, f_1^J, \pi_1^K, z_1^K) = \sum_{k=1}^{K} \chi(f_1^J, q_k). \tag{3.14}$$

$\chi(f_1^J, q_k)$ is a real-valued **decomposed feature function** that calculates the feature we wish to model of the state transition q_k in the context of the original Japanese sentence f_1^J.

Variable k of Equation 3.14 has no meaning in the context of the decoder's beam search, so we write our new features in terms of some general hypothesis q:

$$\chi(f_1^J, q) \tag{3.15}$$

We define a hypothesis (or state transition, if you prefer) q to contain these fields:

q.source : source language phrase

q.target : target language phrase

q.start : index of the first word of the phrase in the source sentence

q.end : index of the last word of the phrase in the source sentence

$q.\text{coverage}_1^J$: $q.\text{coverage}_j = 1$ if the jth source word has been translated, 0 otherwise. (Range $[q.\text{start}, q.\text{end}]$ is covered in q.coverage.)

To give a simple concrete example of this notation, Equation 3.16 gives the definition of the decomposed feature function for the WordPenalty feature (Section 3.4.3), which counts how many words are in the target side of the phrase pair.

$$\chi_{WordPenalty}(f_1^J, q) = q.\text{target.length} \tag{3.16}$$

3.6.1 Pairwise dependency order

These features require the input sentence to have the following annotations:

- words grouped into **chunks**, where a chunk roughly corresponds to a short linguistic phrase.
- part of speech of each chunk
- grammatical case of each chunk
- dependency of each chunk

Chunks are loosely defined; they could be any non-overlapping grouping of contiguous words. In the same way, when we translate Japanese, "words" are loosely defined. As we will see in examples, our Japanese preprocessor (Section 5.2) splits sentences with high granularity into small tokens, often splitting at morpheme boundaries. Still we use "word" to describe each token of Japanese input, even though many

of them could not be considered proper words. If we were to translate from English, it might work well to consider each English word as its own chunk.

We define a chunk x to contain these fields:

x.parent : chunk that this chunk modifies or `NULL`

x.children : list of chunks that modify this chunk

x.pos : part of speech

x.case : case

x.start : index of the first word of the chunk in the source sentence

x.end : index of the last word of the chunk in the source sentence

Chunks are important because they allow our features to consider reordering groups of words together. To illustrate, Gloss 3.3 shows our previous stargazing example (Gloss 3.1) divided into chunks.

(3.3) ⟨私 は⟩ ⟨星 を⟩ ⟨眺め ませ ん でし た⟩ 。
⟨I *TOP*⟩ ⟨stars *Acc*⟩ ⟨gaze at [polite] not [polite] [past tense]⟩ .
"I did not gaze at the stars."

All of the tokens that belong to the verb are grouped in one chunk. The topic marker and accusative case marker (which marks the object) are also grouped together in chunks with their noun. Resulting chunks like ⟨星を⟩ (stars-*Acc*) are called *bunsetsu* (文節) in Japanese grammar. A *bunsetsu* consists of a content word and affixed function words like case markers or verbal morphology [Suzuki and Toutanova, 2006].

One peculiarity to note in Gloss 3.3 is that the period is not in any chunk. Our features gracefully ignore any words that are not in chunks. Furthermore, the dependency structure can consist of multiple subtrees that are not connected. The only restriction that we impose on the dependency graph is that each chunk have at most one parent; that is, each chunk modifies at most one other chunk.

We can now formulate our objective in reordering Gloss 3.3 thusly: irregardless of how the decoder segments the sentence into phrases, we would like as much of the verbal chunk '眺めませんでした' to be translated before the accusative chunk '星を' as possible. To this end, we will define a feature function VERBBEFOREACC that counts up what fraction of the verbal chunk is translated before its accusative modifier.

Example: Defining the VERBBEFOREACC feature

We begin with two indicator functions that identify chunks relevant to the VERBBEFOREACC feature:

$$\text{is\_accusative}(x) = \begin{cases} 1, & \text{if } x.\text{pos} = \text{'Noun' and } x.\text{case} = \text{'Acc'}; \\ 0, & \text{otherwise.} \end{cases} \tag{3.17}$$

$$\text{is\_verb}(x) = \begin{cases} 1, & \text{if } x.\text{pos} = \text{'Verb'}; \\ 0, & \text{otherwise.} \end{cases} \tag{3.18}$$

Then we define two helper functions that compute what fraction of some chunk was already translated before q (Equation 3.19), and what fraction was translated by q (Equation 3.20).

$$\begin{aligned} &\text{frac\_already\_covered}(x, q) \\ &\quad = \frac{\text{num. words in } x \text{ covered in } q.\text{coverage}_1^J \text{ and not in range } [q.\text{start}, q.\text{end}]}{\text{num. words in } x} \end{aligned} \tag{3.19}$$

$$\text{frac\_translated}(x, q) = \frac{\text{num. words in } x \text{ in range } [q.\text{start}, q.\text{end}]}{\text{num. words in } x} \tag{3.20}$$

Finally, we define $\chi_{\text{VERBBEFOREACC}}(f_1^J, q)$, the decomposed feature function for VERBBEFOREACC. For every dependency between a verb and its accusative object in the sentence, Equation 3.21 counts up the fraction of the verbal chunk that has already been translated times the fraction of the accusative chunk is translated by q.

We let X = set of chunks that overlap $[q.\text{start}, q.\text{end}]$ according to dependency parse of f_1^J.

$$\chi_{\textsc{VerbBeforeAcc}}(f_1^J, q) = \sum_{x \in X} \left\{ \begin{array}{l} \text{is\_accusative}(x) \cdot \text{is\_verb}(x.\text{parent}) \\ \cdot \text{frac\_already\_covered}(x.\text{parent}, q) \\ \cdot \text{frac\_translated}(x, q) \end{array} \right\} \tag{3.21}$$

Next we look at a how to compute $\chi_{\textsc{VerbBeforeAcc}}(f_1^J, q)$ for an example hypothesis.

Example: VerbBeforeAcc in action

Consider Figure 3-1, which is the same as Figure 1-1 with part of speech and case annotations. Japanese chunks are separated by spaces.

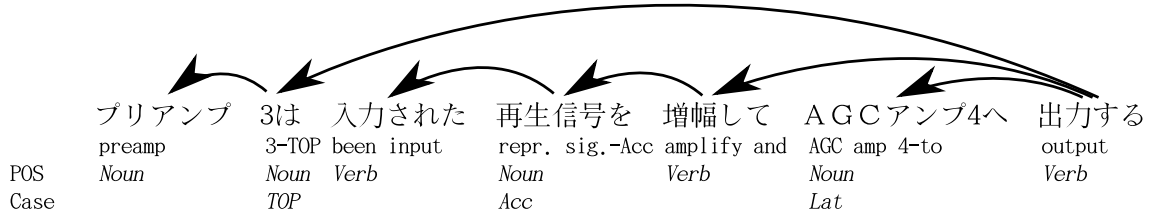

Figure 3-1: Annotated preamp dependency parse.

Now for illustration we will compute the value of VerbBeforeAcc for Hypothesis #318530 first shown in Section 3.5 and repeated here with chunks marked.

Preamp Hypothesis #318530	
Expands	#149442
Covers	⟨プリアンプ⟩ ⟨3 は⟩ ⟨≪入力 さ れ た⟩ ⟨再生 信号≫ を⟩ ⟨増幅 し ...⟩ ⟨... ... ...⟩ ⟨... ...⟩ ...
Phrase pair	≪入力 さ れ た 再生 信号, reproduced signal , which is≫
Features	< VerbBeforeAcc =?, ... >
Score	$-102.947 + \text{future cost} - 102.049 = -204.995$

Notice that q.source covers two chunks on the Japanese side. Its first four words, '入力 さ れ た', completely cover the second chunk, and its last two words, '再生 信号', cover the first two words of the chunk ⟨再生 信号 を⟩.

To compute $\chi_{\text{VerbBeforeAcc}}(f_1^J, q)$, we sum up the contributions of each of the two covered chunks.

- Chunk ⟨入力 さ れ た⟩ has part of speech 'Verb' and no case, which does not match the kind of child we are looking for (a nounal chunk with accusative case), so makes zero contribution.

- Chunk ⟨再生 信号 を⟩ matches the kind of child we are looking for, with part of speech 'Noun' and case 'Acc'. The phrase translated in q, ≪入力 さ れ た 再生 信号≫, covers 2/3 of ⟨再生 信号 を⟩ ("再生 信号" is covered while "を" is uncovered), so frac_translated(⟨再生 信号 を⟩, q) = 2/3.

  Its parent in the dependency tree, ⟨増幅 し て⟩, has part of speech 'Verb' which matches the kind of parent we're looking for. In the hypothesis, '増幅 し' has already been covered, which is 2/3 of the whole chunk ⟨増幅 し て⟩, so frac_already_covered(⟨増幅 し て⟩, q) = 2/3.

  Hence $\chi_{\text{VerbBeforeAcc}}(f_1^J, q) = (2/3)(2/3) = 0.444$.

To compute the value of VerbBeforeAcc for Hypothesis #318530, we add 0.444 to the value of VerbBeforeAcc of the back-linked Hypothesis #149442, which was 0. Therefore the feature vector contains VerbBeforeAcc = 0.444. The positive value indicates that this hypothesis contains a verb coming before its accusative dependency.

General definition

We would like to build a template for features like VerbBeforeAcc so that we can model the dependency orders of other parts of speech and case combinations. In general, we parameterize our pairwise dependency order features on a parameter s

with four fields:

$$s.\text{parent\_pos} : \text{part of speech of parent chunk, or 'Any'}$$
$$s.\text{parent\_case} : \text{case of parent chunk, or 'Any'}$$
$$s.\text{child\_pos} : \text{part of speech of child chunk, or 'Any'}$$
$$s.\text{child\_case} : \text{case of child chunk, or 'Any'}$$

For a given s, we can define two features. The first counts how many times a parent chunk with part of speech s.parent_pos and case s.parent_case is translated before its child with part of speech s.child_pos and case s.child_case. The second counts the opposite: how many times a relevant child chunk is translated before its parent chunk. These two formulations seem redundant, but we found both to be useful when integrated in the decoder.

We begin with two indicator functions in the same vein as is_accusative and is_verb (Equations 3.17 and 3.18) that identify chunks that match the parent or child settings of parameter s.

$$\text{matches\_parent}(x, s) = \begin{cases} 1, & \text{if } x.\text{pos} = s.\text{parent\_pos and } x.\text{case} = s.\text{parent\_case;} \\ 0, & \text{otherwise.} \end{cases} \tag{3.22}$$

$$\text{matches\_child}(x, s) = \begin{cases} 1, & \text{if } x.\text{pos} = s.\text{child\_pos and } x.\text{case} = s.\text{child\_case;} \\ 0, & \text{otherwise.} \end{cases} \tag{3.23}$$

We also reuse the definitions of frac_already_covered and frac_translated in Equations 3.19 and 3.20.

We design $\chi_{\textsc{ParentBeforeChildTemplate}}(f_1^J, q, s)$ in Equation 3.24 to return the sum of the fraction of chunks translated before their children during the translation of

phrase q.source. That is abstruse, but when the decoder sums the contribution from each phrase, it gets the count of chunks (or partial chunks) translated before their children (or partial children).

Again let $X =$ set of chunks that overlap $[q.\text{start}, q.\text{end}]$ according to dependency parse of f_1^J.

$$\chi_{\textsc{ParentBeforeChildTemplate}}(f_1^J, q, s) = \sum_{x \in X} \left\{ \begin{array}{l} \text{matches\_child}(x, s) \cdot \text{matches\_parent}(x.\text{parent}, s) \\ \cdot \text{frac\_already\_covered}(x.\text{parent}, q) \\ \cdot \text{frac\_translated}(x, q) \end{array} \right\} \quad (3.24)$$

To get VerbBeforeAcc, for example, we would instantiate ParentBeforeChildTemplate and set parameters s.parent_pos = 'Verb', s.parent_case = 'Any', s.child_pos = 'Any', and s.child_case 'Acc'.

The second feature ChildBeforeParentTemplate, given in Equation 3.25, is similar to ParentBeforeChildTemplate but counts the opposite ordering: how many chunks are translated before their parents.

$$\chi_{\textsc{ChildBeforeParentTemplate}}(f_1^J, q) = \sum_{x \in X} \left\{ \text{matches\_parent}(x, s) \cdot \sum_{y \in x.\text{children}} \left\{ \begin{array}{l} \text{matches\_child}(y, s) \\ \cdot \text{frac\_already\_covered}(y, q) \\ \cdot \text{frac\_translated}(x, q) \end{array} \right\} \right\} \quad (3.25)$$

Implementation

We take several measures to implement these features efficiently in Moses. Let us say we are translating a sentence with M chunks. First, before translation begins we precompute a map that maps each word position to its chunk index between 0 and $M - 1$. Second, we maintain a vector $chunk\_coverage_1^J$ in each hypothesis where and each $chunk\_coverage_m$ holds the number of words in the mth chunk

that have been translated. These data structures can be updated from the previous hypothesis in time linear in the length of the input sentence, and afford computing $\chi_{\textsc{ParentBeforeChildTemplate}}(f_1^J, q, s)$ and $\chi_{\textsc{ChildBeforeParentTemplate}}(f_1^J, q)$ also in time linear in the length of the sentence.

To integrate these features into Moses, we first need a way to mark up the input sentences with dependency information. We defined a set of tags that can be appended to any word to indicate whether it is a head, what chunk it belongs to, its dependencies, its part of speech, and its case. Then we defined a new input type for Moses called `DependencyTree`, which is a subclass of the default input type `Sentence`. Before translating, `DependencyTree` strips away the dependency annotations and builds an internal representation of the chunks defined in the sentence and their dependency structure. These internal representations can quickly be accessed to compute our feature functions.

Below is the preamp example annotated with its dependencies in `DependencyTree` input format.

```
プリアンプ__head__(0,0)__pos__(n) 3__head__(1,2,0)__pos__(n)__case__(top)
は 入力 さ__head__(3,6)__pos__(v) れ た 再生 信号__head__(7,9,4)__pos__(n)__case__(acc)
を 増幅 し__head__(10,12,8)__pos__(v) て A GC アンプ
4__head__(13,17)__pos__(n)__case__(lat) へ 出力 する__head__(18,19,1,11,16)__pos__(v) 。
```

Notation `信号__head__(7,9,4)__pos__(n)__case__(acc)` indicates that '信号' is head of a chunk that spans the (zero-indexed) 7th to 9th words ('再生 信号 を'), and is modified by the chunk that is headed by the 4th word ('さ').

3.6.2 Chunk cohesion

The motivation for this feature is that a chunk should be translated completely before words from other phrases are interspersed. This feature ChunkCohesion counts up how many chunks have uncovered words remaining when a different chunk is translated. With a negative weight (which we denote $\lambda_{\textsc{ChunkCohesion}}$), it encourages chunks to be translated cohesively without interruption from other chunks. This is

similar to the cohesion feature developed by Cherry [2008], which counted how many times any subtree of the dependency tree was interrupted. Cherry's cohesion feature is complimentary to ours.

Definition

We define a partially covered chunk to be one with at least one uncovered word. We let *previous_partially_covered* be the number of partially covered chunks according to q.coverage$_1^J$ before q.source was translated and *current_partially_covered* be the number of partially covered chunks after q.source was translated.

$$\chi_{\textsc{ChunkCohesion}}(f_1^J, q) = \max(\mathit{current\_partially\_covered} - \mathit{previous\_partially\_covered}, 0) \quad (3.26)$$

Implementation

Similar to the pairwise dependency features, the chunk cohesion feature is easily computed if we maintain a bit vector in each hypothesis that holds whether or not each chunk has any uncovered words.

Example

Here is an example that translates the next phrase in an incohesive way.

Preamp Hypothesis #318478	
Expands	#149442
Covers	⟨プリアンプ⟩ ⟨3 は⟩ ⟨≪入力 さ≫ ... ...⟩ ⟨... ... を⟩ ⟨増幅 し ...⟩ ⟨... ... ...⟩ ⟨... ...⟩ ...
Phrase pair	≪入力 さ, input to a≫
Features	< ChunkCohesion =?, ... >
Score	−103.086 + future cost − 102.990 = −206.077

The back-linked Hypothesis #149442 has two partially covered chunks: ⟨... ... を⟩ and ⟨増幅 し ...⟩. For an expanded hypothesis to incur no cohesion penalty,

it would have to translate some of one of those two chunks. Hypothesis #318478, however, translates ≪入力 さ≫ next and thus adds a third partially uncovered chunk ⟨入力 さ ... ...⟩.

Hence $\chi_{\textsc{ChunkCohesion}}(f_1^J, q) = 3 - 2 = 1$, and the ChunkCohesion feature of Hypothesis #318478 has value 1 as this is the first cohesion violation seen in the path leading to this hypothesis.

3.6.3 Reordering across punctuation

It is often incorrect to translate a word across a punctuation mark, like a comma or quotation mark. This feature Punct counts up how many times a phrase is reordered across a punctuation mark. If its weight $\lambda_{\textsc{Punct}}$ is negative, it discourages reordering across punctuation.

Definition

To calculate this feature, first we let $first\_gap$ be the position of the leftmost uncovered word in q.coverage$_1^J$ before q.source was translated and $next\_punct$ be the leftmost punctuation after $first\_gap$. Hence, in order to not cross punctuation, the next translated phrase must either come completely before $next\_punct$, or include $next\_punct$ and include all uncovered words left of it. The first two regimes of Equation 3.27 express the inverse of these cases.

$$\chi_{\textsc{Punct}}(f_1^J, q) = \begin{cases} 1, & \text{if } next\_punct < q.\text{start}; \\ 1, & \text{if } q.\text{start} \leq next\_punct \leq q.\text{end and } q.\text{start} \neq first\_gap; \\ 0, & \text{otherwise.} \end{cases} \tag{3.27}$$

Implementation

Before beginning beam search in the Moses decoder, we precompute a bit vector p_1^J where p_j is 1 if the jth word of the input sentence is a punctuation mark, and 0

otherwise. Then $first\_gap$ and $next\_punct$ can be computed in time linear to the length of the sentence.

Chapter 4

Reordering before translating

We saw in Chapter 3 that one weakness of phrase-based translation systems is performing the long-distance reordering required when translating from Japanese to English. One way to improve word order in translation output is to reorder Japanese sentences into a more English-like word order in a preprocessing step before translating. Wang *et al.* [2007] recently presented good results performing similar preordering for Chinese→English translation, and others have succeeded with different language pairs [Li *et al.*, 2007; Collins *et al.*, 2005; Kanthak *et al.*, 2005].

We start out by motivating why pre-translation reordering, which we call **preordering**, is a good idea. Then we present two ways to reorder Japanese into an English-like word order. In the first, we split the Japanese at punctuation and the topic marker 'は', then simply reverse the word order of every segment in between. In the second, we use a Japanese dependency parser and several linguistically motivated rules to transform certain Japanese grammatical structures so their surface form has an English word order. Both preorders improved translation quality, as will be shown in Chapter 5.

4.1 Motivation for preordering

If we reorder the Japanese training sentences (and unseen Japanese sentences before translation) into a more English-like word order, we expect a phrase-based system

trained on this new parallel training data to outperform a baseline system trained on the original Japanese sentences. This is because the features used in phrase-based systems (described in Section 3.4) are most effective when not much reordering is required during translation. The preordering step alleviates the need for long-distance reordering during the translation process.

For instance, we noted previously that our Japanese–English word alignments tend to be poor in our baseline Moses system. One plausible advantage of reordering Japanese sentences into a more English-like word order before training the system might be improved word alignment quality. This is because Japanese and English phrases that are translations of each other will be in similar positions, and the word alignment algorithm can safely prefer alignments between words whose position is similar. We expect better word alignment to result in a more accurate phrase table and better word choice in translations.

We implement the following preordering methods in a way that maintains dependency relationships during the reordering. Hence we can output a pseudo-Japanese preorder with dependency annotations that are consistent with the dependencies of the original Japanese sentence.

4.2 Reverse preordering

English is head-initial. Japanese is head-final. So reversing the word order of a Japanese sentence could be a good start towards an English-like order. We factor out the commonality that the topic of English and Japanese sentences both come at the beginning by reversing words before and after the topic marker 'は' separately. Punctuation is kept in the same place.

We begin by tokenizing the sentence with the *Mecab* [Kudo, 2007] morphological analyser, then follow these steps:

1. Split the Japanese sentence at punctuation into a list of "segments".

2. Further split each segment at 'は', the topic marker, to get a pre-topic segment

(which ends with 'は') and post-topic segment. The motivation is that the topic comes at the beginning of both Japanese and English sentences, and should not move to the end.

3. Reverse the order of the words in each segment, so each segment reads backwards.

4. Concatenate the segments and punctuation back together in their original order in the sentence.

We call this reordering the **Rev preorder**. Let us follow these steps to reorder the preamp example, reshown in Gloss 4.1 with words separated by spaces and segment boundaries marked by ‖.

(4.1) プリアンプ 3 は ‖ 入力 さ れ た 再生 信号 を 増幅 し て
Preamp 3-*TOP* ‖ input-*Passive* repr. signal-*Acc* amplify and

A GC アンプ 4 へ 出力 する ‖ 。
AGC amp 4-to output ‖ .
"The preamp 3 amplifies an input reproduction signal, and sends out to an AGC amplifier 4."

The topic segment is 'プリアンプ 3 は', which is reversed into 'は 3 プリアンプ'. The middle segment is also reversed, and these two segments are concatenated together with the final period to get Gloss 4.2, the final Rev preorder.

(4.2) は 3 プリアンプ する 出力 へ 4 アンプ GC A て し 増幅 を 信号 再生 た れ さ 入力 。
TOP-3 preamp output to 4 amp GCA and amplify *Acc*-repr. signal *Passive*-input .

As noted in Chapter 1.5, this Rev preordering could be successfully translated into English monotonically by adding only a few auxiliary words: "The 3 preamp outputs to 4 amp ACG and amplifies the reproduction signal that has been input."

We can analyze this reverse ordering as performing both local and long-distance movement. Long-distance movement can be seen in the verb '出力 する' (output) moving from the end of the sentence to the beginning of the sentence. This long-distance reversal is effective in transforming head-final verb and noun phrases to be

head-initial as they are in English. Local movement can be seen in the verb '出力 され た' (whose tokens are literally, output do [passive] [past tense]) reordering to 'たれ さ 入力' ([past tense] [passive] do output). This local reordering is effective for verbs because most English auxiliaries precede the verb they assist, while Japanese auxiliaries and inflections follow the verb their verb.

This naive REV does have two significant problems. First, subjects marked by 'が', the Japanese subject marker, are reordered to follow their verb. An example of this problem is shown later in Section 4.4.1. We could have chosen to also split segments at 'が', but this would break the word order if the sentence contained a relative clause with 'が' in it. The second problem is that compound nouns are reversed, and English and Japanese compounds already have the same structure. In reversed Gloss 4.2, '再生 信号' (reproduction signal) has been reordered into '信号 再生' (signal reproduction), which is clearly a worse order than the original.

4.3 Dependency tree preordering

In this section we present a more sophisticated way to reorder Japanese into English by flattening a dependency tree parse of the Japanese. We start by running the sentence through *Mecab*, which tokenizes and tags each word with part of speech. We split the sentence into segments at punctuation marks[1], apply our reordering technique to each segment separately, and in the end concatenate the reordered segments and punctuation (in the same order they appeared in the original sentence) together. We call this reordering the **CABOCHA preorder**.

To reorder a segment, we first parse it with the *Cabocha* Japanese Dependency Structure Analyzer [Kudo and Matsumoto, 2002]. The output of Cabocha is a list of **chunks**. These chunks correspond to the notion of chunk we defined in Section 3.6.1: a content word (usually the head) and affixed function words like case markers or verbal morphology. Each chunk contains the following information:

[1] We consider as punctuation marks: 、 , ,。 . ？ ? ！ ! ： : ； ; ＜ ＞ < > （） 『』 【】 〈〉 《》 「」 〜 ~ [] {} ()

- ID number
- Start and end position in sentence
- Chunk that this chunk modifies (in other words, parent chunk)
- Position of head
- Position of the last non-punctuational word

From this list of chunks, we can construct a dependency tree with a node for each chunk and an edge for each dependency. Because of how *Cabocha* constrains its dependency model, all of a node's children precede it in the sentence. As a result, the root node is always the final chunk of the sentence. Figure 4-1 shows the dependency tree constructed from the preamp example (once the period at the end has been split away), with each chunk's head underlined and its part of speech listed. The dependency relations are analogous to those previously shown in Figure 3-1.

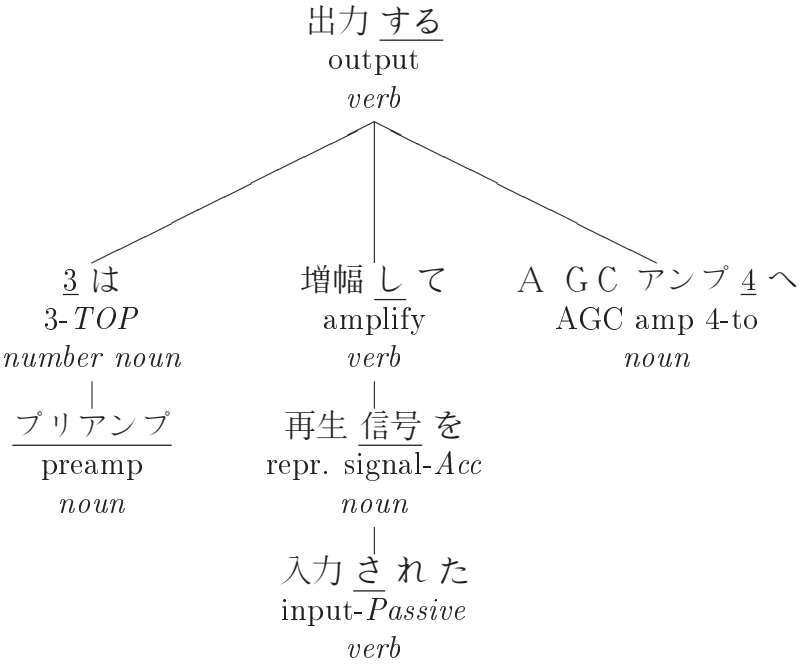

Figure 4-1: Dependency tree for preamp example.

We reorder a Japanese segment in two steps:

1. Flatten the dependency tree according to four rules.

2. Reverse the word order within each chunk.

To flatten the tree we decide for each node into which position among its children to flatten. Algorithm 1 shows the recursive function flatten(*chunk*) that returns an ordered list of chunks containing *chunk* and all of its descendants. The crux of the algorithm is determining where *chunk* should be placed among its children. All non-verbs are placed before their children, which induces a head-initial word order. The placement of verbs is determined by going down the following list:

1. Immediately after rightmost topic or subject, if it exists.
2. Otherwise, immediately before leftmost object, if it exists.
3. Otherwise, immediately after rightmost verb, if it exists. This is to prevent verbs from leapfrogging verbs that preceded them that share only a coordinative dependency.
4. Otherwise, before all children.

We reorder a segment by calling flatten on the root node of its dependency tree, and finally reversing the word order within each chunk. The resulting CABOCHA preorder for our preamp example is shown in Gloss 4.3.

(4.3) は 3　プリアンプ する 出力 て し 増幅　を 信号 再生　た れ さ 入力 へ 4 アンプ ＧＣ Ａ。
TOP-3 preamp output and amplify *Acc*-repr. signal *Passive*-input to 4 amp GCA .

As with the REV preorder, we can add auxiliaries to the gloss of the CABOCHA preorder to form a correct translation: "The 3 preamp outputs the amplified reproduction signals that has been input to 4 amp ACG." The placement of the main verb "output" is questionable; it should probably come after "amplify", with which it coordinates, but our rules put it immediately after its subject, "preamp 3". One fix would be to never place verbs farther left than their leftmost child verb. The verb "amplify" has been placed correctly before its object "reproduction signal". The head-final noun phrase '入力 さ れ た 再生 信号' (input-*Passive* reproduction signal) successfully reordered to be head-initial '信号 再生 た れ さ 入力' (signal reproduction *Passive* input).

Algorithm 1 Calculate flatten(*chunk*) to flatten a Japanese dependency tree into English-like word order

Ensure: *ordered_words* is an ordered list containing *chunk* and all descendants.
  {First, choose where to place *chunk* into its children.}
  pos ⇐ part of speech of head of *chunk*
  ordered_children ⇐ list of children of *chunk*, ordered as they were in original sentence
  if *pos* = 'Verb' and a child has a subject marker 'が' or topic marker 'は' **then**
    insert *chunk* into *ordered_children* after rightmost child subject or topic
  else if *pos* = 'Verb' and a child has an object marker 'を' **then**
    insert *chunk* into *ordered_children* before leftmost child object
  else if *pos* = 'Verb' and a child has head with part of speech 'Verb' **then**
    insert *chunk* into *ordered_children* after rightmost child verb
  else
    insert *chunk* into beginning of *ordered_children*.
  end if
  {Second, recursively flatten each child.}
  ordered_words ⇐ [ ]
  for all *child* in *ordered_children* **do**
    if *child* = *chunk* **then**
      append *child* to *ordered_words*
    else
      append flatten(*child*) to *ordered_words*
    end if
  end for

Thanks to its systematic head-final to head-initial inversion, we found that the CABOCHA preorder tended to closely match English word order. We demonstrate in Section 5.5 that CABOCHA dominates REV and BASELINE (no reordering) preorders in translation quality when translating monotonically (that is, not allowing reordering other than what has already been reordered in the preorder). We will now take a look at examples of CABOCHA and REV preorders and what it looks like to translate them monotonically.

4.4 Preorder examples

On the next two pages, we present a pair of example sentences reordered into REV and CABOCHA preorder. Each example shows a gloss of BASELINE, REV, and CABOCHA preorders. Under each gloss is the *monotonic* translation of the preorder (using Moses trained on equivalently preordered data with its baseline set of feature functions). The first example 4.4.1 causes hiccoughs for both REV and CABOCHA, and the second example 4.4.2 is reordered correctly to the same word order with both preordering methods.

4.4.1 Example: Thwarted by lock release pins

Reference: The lock release pin is set to a longitudinal length so that it does not abut against the front wall inner surface of opening 26.

BASELINE .....

(4.4) ロック 解除 ピン が 開口 26 の 前端 壁 内面 に 当 接
Lock release pin *Nom* opening 26 *Gen* front end wall inner surface to abut

し ない よう な 前後 方向 の 長 さ に 設定 さ れ て いる 。
do not way of longitudinal direction *Gen* long -ness to set *Passive Present*.

"The lock is released until when the lock release pin opening 26 of the front wall abut against the inner surface such that in a front-to-rear direction."

The monotonic translation has no chance to preserve the meaning of the original sentence because the word order is unsalvageable.

REV .....

(4.5) いる て れ さ 設定 に さ 長 の 方向 前後
Present Passive set to -ness long *Gen* direction longitudinal

な よう ない し 接 当 に 内面 壁 前端 の
of way not do abut to inner surface wall front end *Gen*

26 開口 が ピン 解除 ロック 。
26 opening *Nom* pin release lock .

"Is set at a length of the longitudinal direction so as not to abut against the inner surface of the front end wall of the opening 26 and a pin is unlocked."

The subject of the sentence, "release lock", is reordered to the end of the sentence, as is a forementioned systematic problem with the REV method.

CABOCHA .....

(4.6) いる て れ さ 設定 に さ 長 の 方向 前後
Present Passive set to -ness long *Gen* direction longitudinal

が ピン 解除 ロック な よう ない し 接 当 に 内面
Nom pin release lock of way not do abut to inner surface

壁 前端 の 開口 26 。
wall front end *Gen* opening 26 .

"Is set to a length of the longitudinal direction of the release pin locked so as not to abut against the inner surface of the front end wall of the opening 26."

Excellent, except that "release lock" should appear at the beginning of the sentence. The problem is that "release lock" needs to be both the subject of "abut" and the object of the passive "set", but appears only once in the Japanese sentence. The CABOCHA preorder algorithm, based on the dependency tree, chooses to put "release lock" in the subject position of "abut" instead of "set", which would work better here.

4.4.2 Example: Smooth clockings

Reference: Register 35 has a function of delaying the signal Not Taken for 1 clock cycle.

BASELINE .....

(4.7) レジスタ 35 は 、 信号 NOTTAKEN を 1 クロック 遅延 さ せる 機能 を 備える 。
Register 35 *TOP* , signal Not Taken *Acc* 1 clock delay *Causative* function *Acc* provide .
Register 35 comprises a signal NOTTAKEN a delay of one clock period .

REV .....

(4.8) は 35 レジスタ 、 備える を 機能 せる さ 遅延 クロック 1 を NOTTAKEN
TOP 35 register , provide *Acc* function *Causative* delay clock 1 *Acc* Not Taken
信号 。
signal .
"Register 35 has a function of delaying one clock predicts NotTaken signal."

The monotonic translation is good, with exception that "predicts" has been strangely inserted.

CABOCHA .....
(The preorder is identical to REV.)

(4.9) は 35 レジスタ 、 備える を 機能 せる さ 遅延 クロック 1 を NOTTAKEN
TOP 35 register , provide *Acc* function *Causative* delay clock 1 *Acc* Not Taken
信号 。
signal .
"Register 35 is provided with a function for delaying one clock predicts NotTaken signal."

Even though the preorder has correct word order, the monotonic translation is poor because "provide" is needlessly made passive.

Chapter 5

Experiments

This chapter describes the setup of our Japanese→English Moses system and the experiments we performed with it to measure the effectiveness of the new feature functions presented in Chapter 3 and the reordering preprocessors of Chapter 4. Overall, our best system combined a tuned selection of feature functions with our reverse preprocessor to increase BLEU score 27.96→28.74.

5.1 Training data

Phrase-based translation systems require a large corpus of parallel text to build their translation model, and the larger the corpus, the higher translation quality. Fortunately, Masao Utiyama has spearheaded creation of two very large parallel Japanese–English corpora in the patent and news domains. Our system is trained on 58.6 million words (measured on the English side) of parallel text, 53.5 million of which are patent data. The training corpus includes:

- Japanese-English Patent Parallel Corpus [Utiyama *et al.*, 2007] training set provided for the NTCIR-7 Patent Translation Task [Fujii *et al.*, 2007], 53.5 million words of Japanese–English patent data.

- Japanese-English News Article Alignment Data [Utiyama and Isahara, 2003], 3.6 million words from the Yomiuri Shimbun and Daily Yomiuri newspapers.

- Tanaka Corpus [Tanaka, 2001], 1.2 million words of sentences collected by Yasuhito Tanaka's students.

- EDICT Japanese-English Dictionary [Breen, 1995], 0.45 million words from a general-use dictionary.

We trained a 5-gram language model on only the English side of the Patent Parallel Corpus training set. We use the 915-sentence development (`dev`) and 899-sentence test (`test`) sets, both single-reference, supplied for the NTCIR-7 Patent Translation Task [Utiyama *et al.*, 2007]. These 1814 sentences were held out from the Patent Parallel Corpus training set but come from the same collection of patents.

5.2 Preprocessing

Japanese is written without spaces, so we use the *Mecab* [Kudo, 2007] morphological analyser to tokenize the Japanese data (add spaces between words). We further tokenize punctuation using the Moses script `tokenizer.perl`. We normalize wide-character numbers to their ASCII (the patent data contain many wide-character numbers) and discard sentences longer than 100 words. As is the recommended setup for Moses systems, we lowercase all English words during preprocessing, and recase words as a postprocessing step using a recaser provided with Moses [WMT Baseline, 2007].

When preprocessing development or test data for translation by Moses, the final step is to annotate each sentence in our `DependencyTree` input format described in Section 3.6.1 so that the decoder can read it as input.[1] The chunking, dependency

[1]Moses already supports annotation of the input in two forms that we did not use in these experiments. The first is its flagship "factored translation" capability, where one can translate not just surface form but also build phrases with part-of-speech, stemmed form, or other factors. In preliminary experimentation, we found that using *Mecab* part of speech as an factor did not lead to a significant BLEU increase.

The second advanced Moses feature is "confusion net" decoding, where one can pass multiple candidates for each source word as input, could be more useful. Dyer [2007] translates confusion nets wherein each word has its surface form and various stemmed forms as candidates, and found that this improved quality when translating from morphologically complex languages. This technique could improve translation of rarely-seen conjugations of Japanese verbs.

parse, and part of speech tags are the result of the process described in Section 4.3. Each chunk's grammatical case is determined by looking up the last word in the chunk and its part of speech in Table 5.1. An explanation of each case will be given below in Section 5.6

Last word and POS		Case	Abbreviation
が	助詞-格助詞	Nominative	'Nom'
は	助詞-格助詞	Topic	'TOP'
を	助詞-格助詞	Accusative	'Acc'
の	助詞-連体化	Genitive	'Gen'
へ	助詞-格助詞	Lative	'Lat'
に	助詞-格助詞	Dative	'Dat'
Anything else			'None'

Table 5.1: Select Japanese postpositions and the case they mark.

5.3 Automatic evaluation metrics

We use **BLEU score** on our `test` corpus to evaluate translation quality of our baseline and modified Moses systems. Designed by Papineni *et al.* [2001], BLEU is ubiquitously used to compare machine translation output across systems and is the official evaluation metric for the NIST and NTCIR machine translation evaluations.

BLEU compares machine translation output to reference translations. The more similar they are, the higher the score, which ranges from 0 to 100. Similarity is measured by **n-gram precision**; the more words, bigrams, trigrams, and 4-grams from a translation that appear in the reference, the better. Because n-gram precision does not directly model long-distance word order, it is unclear whether or not BLEU can account for differences in word order between translations [Callison-Burch *et al.*, 2006].

Lavie and Agarwal [2007] introduced another automatic evaluation metric called METEOR, which, unlike BLEU, explicitly accounts for the alignment between matching words of the reference and the translation. One component of METEOR is the **fragmentation score**, which is a measure of how *dissimilar* the order of the words

that match in both the translation and the reference are. The *lower* the METEOR fragmentation score, the *better* the word order.

We include plots of both BLEU and METEOR fragmentation scores when discussing our results. We defer an in-depth discussion of BLEU and METEOR to Section 5.13, where we analyze whether these metrics are capable of capturing differences in word order between our system.

5.4 Experiments with decoder parameters

The most important Moses decoder parameter is maximum distortion limit, which we denoted $MaxDistortion$ and described in Section 3.5.1. The larger the $MaxDistortion$, the higher the freedom for phrases to move around during transltion. Table 5.2 shows BLEU score decoding with a range of $MaxDistortion$ settings, different preorders, and the baseline Moses feature functions listed in Section 3.4 with weights tuned for $MaxDistortion = 6$.[2] (Setting $MaxDistortion = -1$ corresponds to unlimited reordering.)

Table 5.2 verifies that when the language pair has very different word order, long-distance reordering is crucial for high translation quality. When translating preorder REV, which has a roughly English word order, quality peaks at about $MaxDistortion = 9$, and drops off for higher values. In contrast, when translating the BASELINE (no reordering) preorder, the higher the setting of $MaxDistortion$, the higher the translation quality. We can interpret this result as follows: Translating between REV and English, most words need to move fewer than 6 places, so allowing them to move farther results in incorrect reordering; translating between BASELINE and English, some words need to move farther than 9 places, so disallowing such long movement rules out many correct translations.

Based on these results, in later experiments we set $MaxDistortion = 9$ unless otherwise noted.

Table 5.3 shows BLEU score for a $MaxDistortion = 9$ REV system decoding with

[2] This system used a non-patent recaser, so scores are not directly comparable with other systems.

MaxDistortion	BASELINE	REV	CABOCHA	Seconds per sentence[3]
0	20.86	20.32	21.61	2.2
6	23.76	25.44	24.79	5.0
9	25.24	25.49	25.12	7.8
-1	26.07	25.08	24.58	37.2

Table 5.2: How *MaxDistortion* affects BLEU score and translation time for different preorders.

various stack size settings, which controls the beam width in the the decoder's beam search.[4] A larger beam width means fewer search errors are made. The Moses default is 100, and these results show that increasing it does not significantly improve quality. Because translation is much slower with a large stack size, we use the default 100 in our experiments.

Stack size	BLEU	Seconds per sentence
100	28.46	4.5
200	28.63	8.4
400	28.51	16.1

Table 5.3: How stack size affects BLEU score and translation time.

5.5 Evaluating preorder efficacy

Table 5.2 also illustrates the impact of preordering on translation quality. When no reordering is allowed during decoding, CABOCHA achieves the highest BLEU score, validating our observation that its word order is closest to English. However, with a limited amount of reordering, REV is the leader. This is a very surprizing result, but one that was consistent across test corpora or feature function choice.

Equally surprizing is that when unlimited reordering is allowed, the BASELINE preorder, which is the original Japanese word order, performs best. This is shocking, and we can offer no explanation. With unlimited reordering and employing the

[4]This system was trained only on the patent corpus, so scores are not directly comparable with other systems. It is notable that our systems trained on only on the patent data tended to outperform equivalent systems trained on our full training data (consisting of patent data, news data, and dictionaries) in experiments on our patent-domain `test` corus.

default Moses feature functions, only the language model can evaluate long-distance reorderings. Because language model scores are in no way conditioned on the source sentence, the language model cannote advise the decoder on how to reorder words.

Without the feature functions we developed in Section 3.6, the decoder is "driving blind" when positioning words far away from their original spot, but has maximum freedom to assemble them according to the language model into fluent English, which leads to a high BLEU score. Still, we would expect one of the preordered systems to outperform the baseline. It may be the case that the phrase table of the baseline system is unexpectedly of higher quality than that of the preordered systems, or that the local inversion in the preordered systems degrades BLEU score with unlimited reordering. We continue to compare Rev versus Baseline as the preorder for upcoming experiments.

5.6 Long-distance reordering features

Now we turn our attention to the experiments with the long-distance reordering feature functions we incorporated into Moses in Section 3.6.1.

First we consider two general features instantiated from ParentBeforeChildTemplate and ChildBeforeParentTemplate with parameter s set to 'Any' for all fields, so that it tracks the order of every pairwise dependency. We get feature ParentBeforeChild, which counts how often a parent is translated before its child, and feature ChildBeforeParent, which counts how often a child is translated for its parent. Feature ParentBeforeChild should encourage more translations with parents ordered before their children when we set its weight, denoted $\lambda_{\text{ParentBeforeChild}}$, to be positive. When given a negative weight, feature ChildBeforeParent should encourage similar behavior.

To test our features, we trained 'Baseline' and 'Rev' preordered systems, tuned their weights using $MaxDistortion = 6$, and normalized all weights so their absolute values sum to 1. We use these systems as the baseline. For each feature, we redecoded the `dev` and `test` corpora with its weight set to a range of values spaced every 0.05

over the interval where the feature appeared useful. We report the weight that led to maximum score on the `dev` corpus, this maximum `dev` score, and the `test` corpus score using that weight. In Table 5.4, we show the results for ParentBeforeChild and ChildBeforeParent. Both give small improvements. Plots of `test` scores using these general features are given in Figures 5-1 and 5-2.

Feature	Weight	`dev` BLEU	`test` BLEU
ParentBeforeChild	0.25	+0.13	+0.39
ChildBeforeParent	-0.30	+0.09	+0.08

Table 5.4: Best scores for general pairwise features.

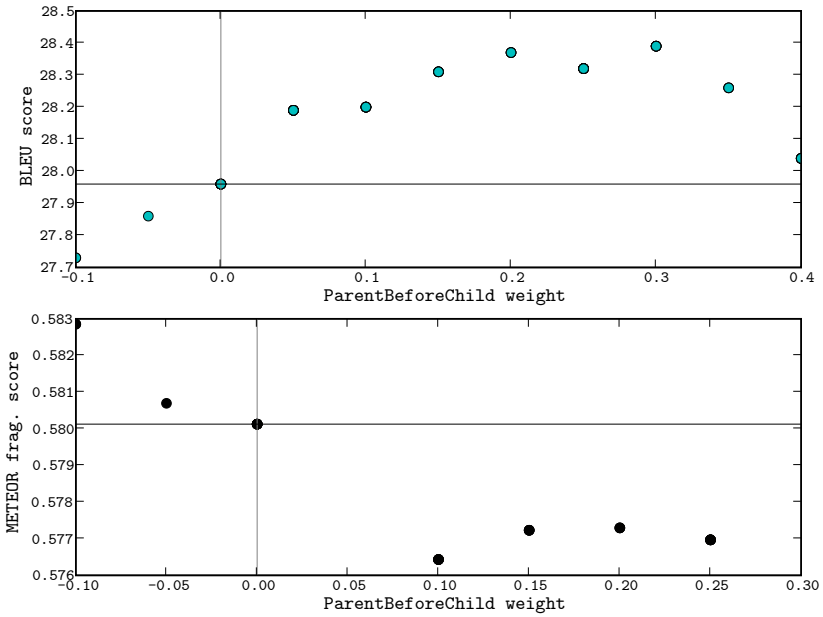

Figure 5-1: $\lambda_{\text{ParentBeforeChild}}$ against BLEU and METEOR fragmentation scores with Baseline preorder.

The difference between these two general pairwise dependency order features is that ParentBeforeChild should upgrade the score of translations with better word order, and ChildBeforeParent should downgrade the score of translations with worse word order. Evaluating which of these approaches will be more effective in the decoder is very difficult, so we experimented with both. Table 5.4 suggests that

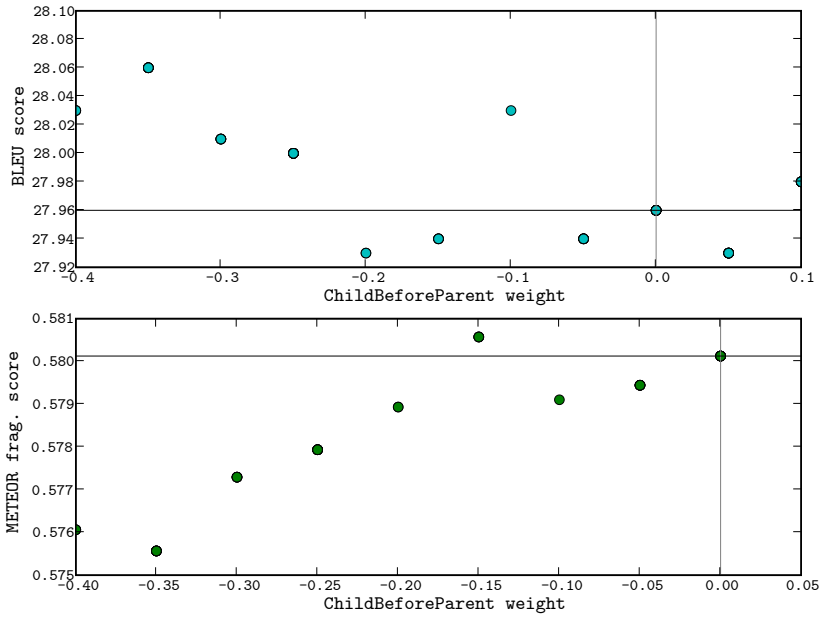

Figure 5-2: $\lambda_{\text{ChildBeforeParent}}$ against BLEU and METEOR fragmentation scores with Baseline preorder.

ParentBeforeChild is more effective, and that promoting correct translations is more useful than demoting incorrect ones.

Next we will analyze more specific features. While ParentBeforeChild and ChildBeforeParent track every dependency, we now define features that track only how dependencies between words with certain parts of speech and case are ordered. We start off with these three features that capture the biggest differences in Japanese and English word order:

VerbBeforeAcc counts when verbs come before their object. (ParentBeforeChildTemplate with s.parent_pos = 'Verb', s.child_case = 'Acc')

NounBeforeGenTemplate counts when nouns come before a genitive noun that modifies them. (ParentBeforeChild with s.parent_pos = 'Noun', s.child_case = 'Gen')

NounBeforeVerbTemplate counts when nouns come before the verb of a relative clause that modifies them. (ParentBeforeChild with s.parent_pos =

'Noun', s.child_pos = 'Verb')

The performance of these features are summarized in Table 5.5. All three features improved translation quality, and NounBeforeGen led the pack with a +0.38 BLEU improvement. Plots of `test` scores using these specific features are given in Figures 5-3–5-5.

Feature	Weight	`dev` BLEU	`test` BLEU
VerbBeforeAcc	0.30	+0.06	+0.05
NounBeforeGen	0.25	+0.25	+0.38
NounBeforeVerb	0.25	+0.05	+0.12

Table 5.5: Best scores for specific pairwise features.

Now we will show examples of each feature at work.

5.6.1 Verb before accusative argument feature

Feature VerbBeforeAcc successfully fulfilled its goal of encouraging translations with correct English Subject–Verb–Object order. Although the BLEU score increase is a miniscule, many translations improve to a better word order. See Figure 5-3 In the following example, "serve" moves to before its object (in the Japanese sentence) "guide for the moving holder 3".

Japanese	７はシル材であり、後述の加工ガス９のシルと移動ホルダ３のガイドを兼ねたものである。
Reference	A sealant 7, which serves as a seal for cutting gas 9, also serves as a guide for the moving holder 3.
$\lambda_{\text{VerbBeforeAcc}} = 0$	7 is a seal material of the working gas 9 seal and the moving holder 3 also serves as a guide.
$\lambda_{\text{VerbBeforeAcc}} = 0.40$	7 is a seal member for sealing the machining gas 9 and also serves as a guide for moving holder 3.

The Moses baseline system translates many sentences into English sentences with passive main verbs, because this is the most natural way to construct a verb-final English sentence if the verb is not motivated to reorder to earlier in the sentence. The VerbBeforeAcc feature correctly activizes some of these passive sentences:

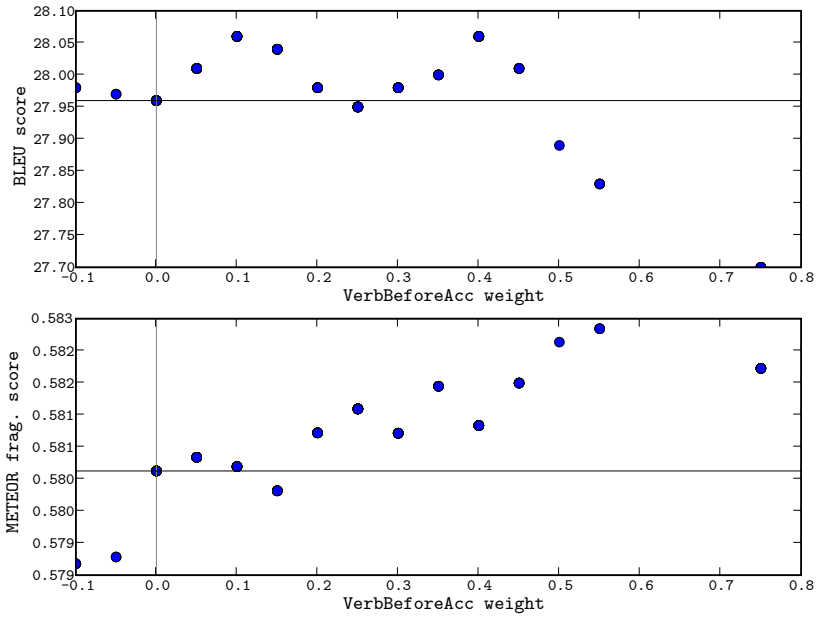

Figure 5-3: $\lambda_{\text{VERBBEFOREACC}}$ against BLEU and METEOR fragmentation scores with BASELINE preorder.

Japanese	そして、ビットマップデータ生成部３９は、各色毎の濃度に応じて、これら各色毎のビットマップデータを生成する。
Reference	Then, the bit map data generating section 39 generates bit map data for each color according to each color density.
$\lambda_{\text{VERBBEFOREACC}} = 0$	Then, the bit map data generator 39 according to the density of each color, the bit map data of each color is generated.
$\lambda_{\text{VERBBEFOREACC}} = 0.40$	Then, the bit map data generator 39 according to the density of each color, and generates bit map data for each of the colors.

Naturally, even if the verb successfully moves before its object, the translation might not improve. Here, “generated” moves before its object “braking torque”, but the sentence remains passive and incomprehensible.

Japanese	そして、ロータ１６とステータ１５との間に充填した液体の運動エネルギーが熱エネルギーに変換されて制動トルクを発生する。
Reference	And, the kinetic energy of the liquid filled between the rotor 16 and stator 15 is converted into thermal energy to thereby produce a brake torque.
$\lambda_{\text{VERBBEFOREACC}} = 0$	Then, the rotor 16 and between the stator 15 of the liquid to be filled in the kinetic energy is converted to thermal energy braking torque is generated.
$\lambda_{\text{VERBBEFOREACC}} = 0.40$	Then, the rotor 16 and between the stator 15 of the liquid to be filled in the kinetic energy is converted to thermal energy generated by the braking torque.

Overall, we think VERBBEFOREACC improves translation quality more than the small BLEU score improvement indicates. It causes translations to better preserve the meaning of the original sentence, and has no observable systematic negative effect.

5.6.2 Noun before genitive modifier feature

Feature VERBBEFOREACC earned the largest BLEU increase of our features, as shown in the BLEU Figure 5-4. It aims to translate the Japanese pattern 'A の B' into "B of A" by encouraging noun B to move before the genitive-case noun A.[5] Examples include 'アメリカの大統領' (literally, America-*Gen* president) to "President of the United States", '田中のお父さん' ('Tanaka-*Gen* father') to "father of Tanaka", and '世界の窓' (world-*Gen* window) to "window to the world". However, just as often, 'A の B' can be translated without swapping A and B; examples include '私の論文' (I-*Gen* thesis) to "my thesis", '茶色の本' (brown-*Gen* book) to "brown book", or translating the first two examples as "United States President" and "Tanaka's father".

Although NOUNBEFOREGEN achieved a significant BLEU score increase, it is

[5] 'の' is not usually considered a case marker, but instead a conjunctive particle indicating adnominal relation [Suzuki and Toutanova, 2006]. For our purposes, however, it is beneficial to think of 'の' as marking the preceding noun phrase with genitive case, which means that it modifies the following noun phrase. 'の' is pronounced like the Japanese dramatic style noh, and functions similarly to Chinese 的 ('de' in pinyin).

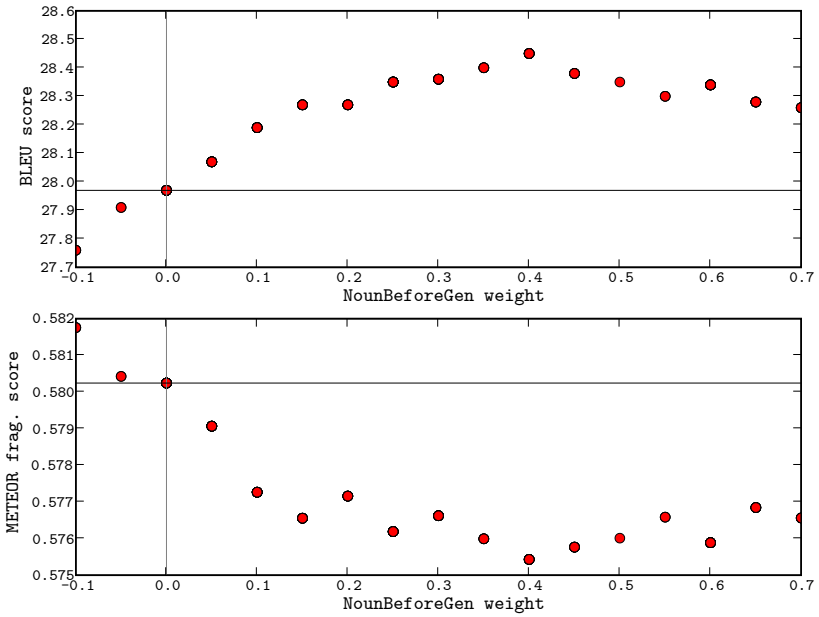

Figure 5-4: $\lambda_{\text{NounBeforeGen}}$ against BLEU and METEOR fragmentation scores with Baseline preorder.

harder to find instances where it subjectively improved translation quality compared to experiments with VerbBeforeAcc. Many sentences are randomly affected and are often reordered into "sound bites" that increase BLEU score without increasing translation quality.

Nevertheless there are some examples of clear improvements in noun phrase reordering. In the following example, '命令キュー１３の状態' (instruction queue 13-*Gen* state) correctly becomes "state of the instruction queue", whereas before "instruction" was dropped.

Japanese	図１４は分岐命令が実行されたサイクルにおける命令キュー１３の状態を示す。
Reference	FIG. 14 shows one example of the state of queue 13 in the cycle in which a branch instruction is executed.
$\lambda_{\text{NounBeforeGen}} = 0$	FIG. 14 is a branch instruction is executed in a cycle of the instruction queue 13.
$\lambda_{\text{NounBeforeGen}} = 0.40$	FIG. 14 is a branch instruction is executed in a cycle shows the state of the instruction queue 13.

Similarly, this example correctly forms "implantation of impurity ions".

Japanese	この工程においては、ソース／ドレイン領域を形成するための条件で不純物イオンの注入を行う。
Reference	In this step, impurity ions were implanted for forming the source and drain regions.
$\lambda_{\text{NounBeforeGen}} = 0$	In this process, since the source / drain region is formed under the conditions of the impurity ions are implanted.
$\lambda_{\text{NounBeforeGen}} = 0.40$	In this process, since the source / drain region is formed under the conditions of implantation of impurity ions.

This next example is translated correctly with or without reordering the arguments "buffer counter" and "initial value" of 'の'.

Japanese	バッファカウンタの初期値はNに設定される。
Reference	The initial value of the buffer counter is set to N.
$\lambda_{\text{NounBeforeGen}} = 0$	The buffer counter is set to the initial value N.
$\lambda_{\text{NounBeforeGen}} = 0.40$	The initial value of the buffer counter is set to N.

Finally, here is an example of a noisily affected translation. Here 'の' is used as part of fixed grammatical construct ('Aの方が優れている', "A is better"), so reordering its arguments is not desirable. With the NounBeforeGen-induced reordering, the meaning of the second clause is lost, but the word choice is flukily better (it includes "better", which also appears in the reference), so mistranslating this example might boost BLEU score.

Japanese	ここで、図１０と図１２とを比較すれば、図１０の特性の方が優れている。
Reference	Comparing FIGS. 10 and 12 indicates that the characteristics shown in FIG. 10 are better than those in FIG. 12.
$\lambda_{\text{NounBeforeGen}} = 0$	In this case, the comparison between FIGS. 10 to 12, the characteristic is more excellent in FIG. 10.
$\lambda_{\text{NounBeforeGen}} = 0.40$	In this case, the comparison between FIGS. 10 to 12, it is better characteristics of FIG. 10.

5.6.3 Noun before verbal modifier

In Japanese, relative clauses precede the noun they modify. For instance, '叫んでいる男' (is yelling man) means "man who is yelling". The NounBeforeVerb feature gives high marks to translations that reorder relative clauses to follow the nouns that they modify. The impact on BLEU score, shown in Figure 5-5, was small, but we believe that the positive slope when $0 \leq \lambda_{\text{VerbBeforeAcc}} \leq 0.30$ indicates that $\lambda_{\text{VerbBeforeAcc}}$ does have a significant positive effect on translation quality.

In the following improved example, NounBeforeVerb causes '参照光が伝播する光路長' (reference light-*Nom* propagate optical path length) to correctly reorder into "optical path length of the light propagating on a reference", which is very close to the reference translation "optical path length through which the reference light is propagated".

Japanese	このことにより、参照光が伝播する光路長を変化させることができる。
Reference	As a consequence, the optical path length through which the reference light is propagated may be varied.
$\lambda_{\text{NounBeforeVerb}} = 0$	As a result, the reference light is propagated through the optical path length can be changed.
$\lambda_{\text{NounBeforeVerb}} = 0.30$	As a result, the optical path length of the light propagating on a reference can be changed.

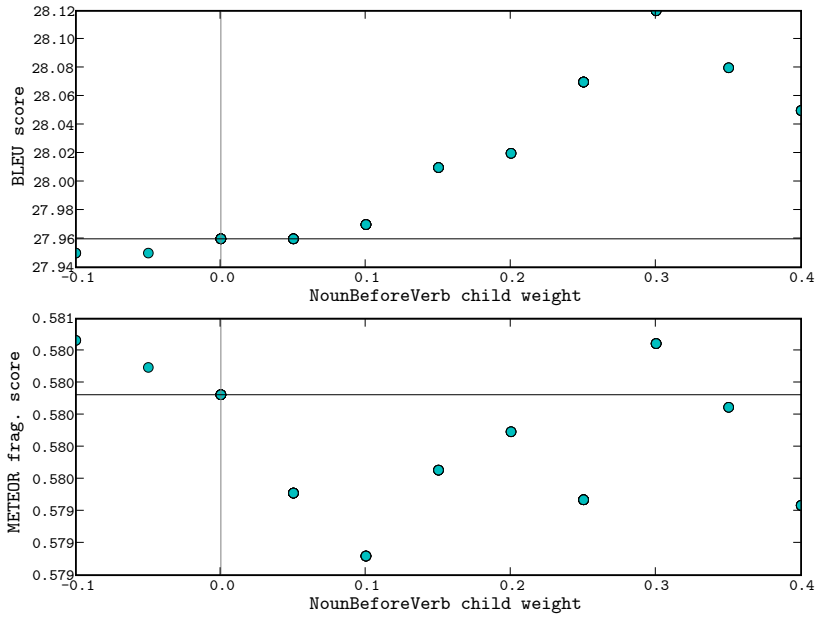

Figure 5-5: $\lambda_{\text{NounBeforeVerb}}$ against BLEU and METEOR fragmentation scores with Baseline preorder.

Relative clauses are especially pervasive in Japanese grammar. Here, "after" follows relative clause "relief valve 140 operates", which NounBeforeVerb successfully reorders to the beginning of the sentence to form "After the relief valve 140 is operated..."

Japanese	リリーフ弁１４０が作動した後の操舵力は、マニアル操舵特性に平行な直線となる。
Reference	Upon the operation of the pilot relief valve 140, the steering force is defined by lines which are in parallel indicating the manual steering characteristic.[6]
$\lambda_{\text{NounBeforeVerb}} = 0$	The relief valve 140 is operated to the steering force after the straight line in parallel with the manual steering characteristic.
$\lambda_{\text{NounBeforeVerb}} = 0.30$	After the relief valve 140 is operated to steering force is a straight line parallel to the manual steering characteristic.

[6]The given reference translation is a stand-in to make this example easier to understand. The

Naturally, there are plenty of sentences that unexplainably changed for the worse. Here is one, where the baseline's correct "number of bits" turns into "bit number".

Japanese	加算により得られるデータ、すなわちシリアルアドレスは、最終的に決定される総ビット数よりも小さな値に選ばれなければならない。
Reference	Data obtained by addition, that is, the serial address, must be chosen to be a value smaller than the finally determined total number of bits.
$\lambda_{\text{NounBeforeVerb}} = 0$	The data obtained by the addition, that is, the serial address is determined to be a value smaller than the total number of bits must be selected.
$\lambda_{\text{NounBeforeVerb}} = 0.30$	The data obtained by the addition, that is, the serial address is determined to be a value smaller than the total bit number must be selected.

5.7 Feature performance with unlimited reordering

We hypothesized that our long-distance reordering features might offer more improvement if the decoder allowed unlimited reordering. To test this, we decoded the `test` corpus setting $MaxDistortion = -1$ with a range of values for features VerbBeforeAcc, NounBeforeGen, and NounBeforeVerb. Table 5.6 shows the maximum BLEU score achievable with the perfect weight for the `test` corpus. The features are beneficial with either limited and unlimited reordering.

Figures 5-6–5-8 plot the performance of these three features decoding `test` with different weights and no distortion limit. NounBeforeGen gives markedly less possible benefit over the baseline with $MaxDistortion = -1$ compared to with $MaxDistortion = 9$ (which was shown in Figure 5-4). NounBeforeVerb in contrast offers a larger improvement with more reordering allowed.

original reference included many things not mentioned in the Japanese sentence: "Upon the operation of the pilot relief valve 140, the steering force is defined by one of four thin lines which are in parallel with the thick line indicating the muscular-energy steering characteristic." The problem of creative reference translations is an issue with any test corpus.

Feature	$MaxDistortion = 9$	$MaxDistortion = -1$
VerbBeforeAcc	+0.14	+0.15
NounBeforeGen	+0.48	+0.26
NounBeforeVerb	+0.16	+0.30

Table 5.6: Maximum BLEU improvements on `test` corpus for limited and unlimited reordering.

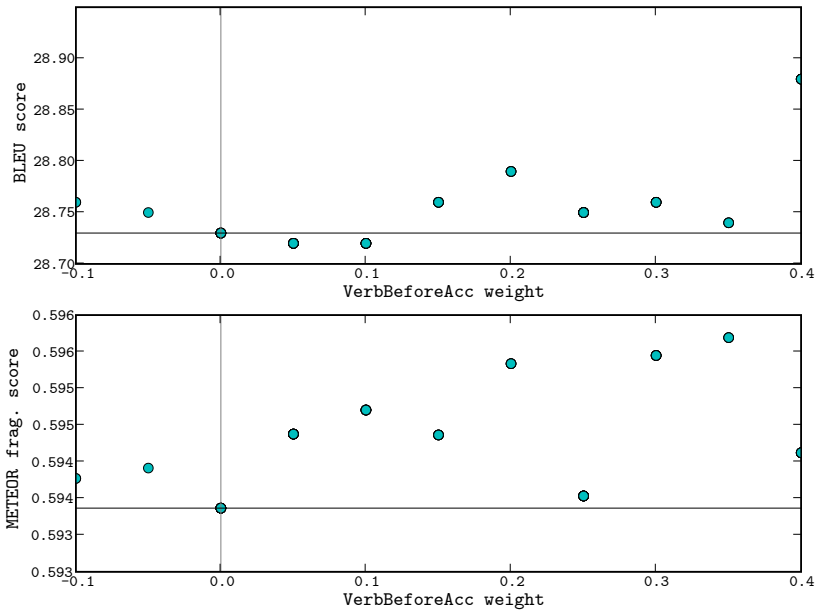

Figure 5-6: $\lambda_{\text{VerbBeforeAcc}}$ against BLEU and METEOR fragmentation scores with Baseline preorder and unlimited reordering.

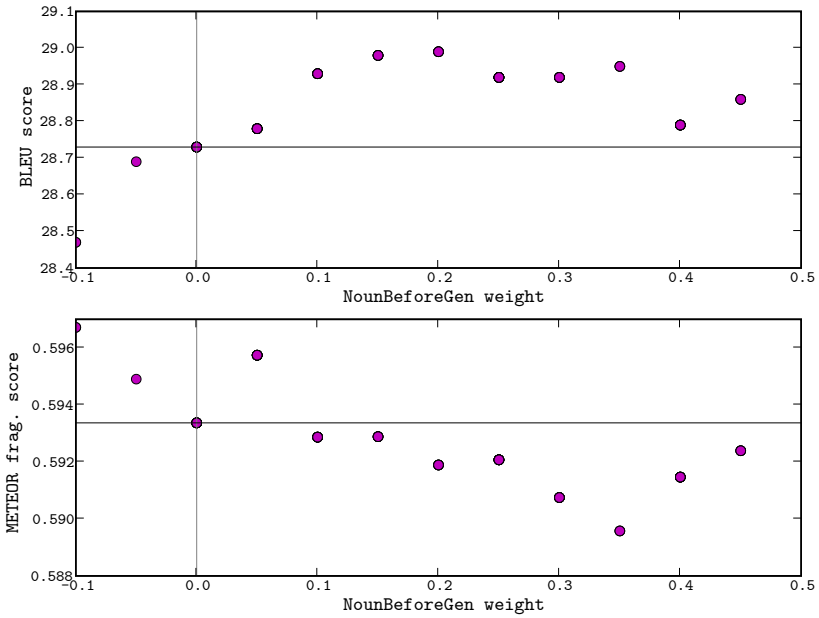

Figure 5-7: $\lambda_{\text{NounBeforeGen}}$ against BLEU and METEOR fragmentation scores with Baseline preorder and unlimited reordering.

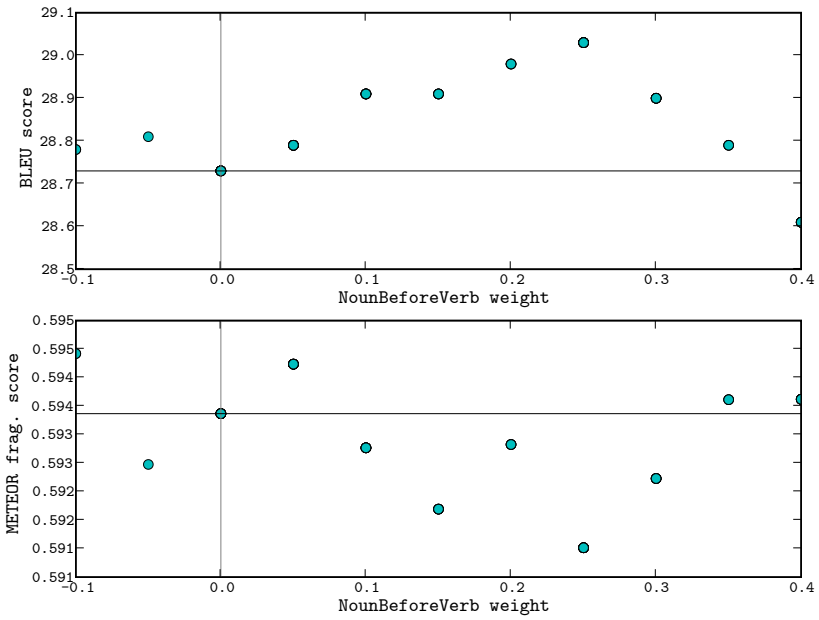

Figure 5-8: $\lambda_{\text{NounBeforeVerb}}$ against BLEU and METEOR fragmentation scores with Baseline preorder and unlimited reordering.

5.8 Feature performance on REV preorder

Because our long-distance reordering features focus on target-side word order, we would expect the same features to be useful regardless of preorder. On the contrary, we found that our long-distance reordering features were largely useless when decoding preordered sentences. Table 5.7 shows the performance across features. We can infer that when translating with the REV preorder, the decoder does not need our feature functions to guide word order, because long-distance is unnecessary to translate preordered sentences.

Feature	Weight	`dev` BLEU	`test` BLEU
PARENTBEFORECHILD	0.05	+0.02	-0.13
CHILDBEFOREPARENT	-0.50	+0.09	+0.14
VERBBEFOREACC	0.15	+0.08	+0.07
NOUNBEFOREGEN	0.15	+0.01	-0.04
NOUNBEFOREVERB	0.15	+0.04	-0.05

Table 5.7: Best scores with REV preorder.

The mystery in Table 5.7 is why CHILDBEFOREPARENT is more useful than PARENTBEFORECHILD when decoding preordered sentences, while we saw the opposite pattern when decoding BASELINE preorder. This result may be attributable to noise. Figures 5-9 and 5-10 show BLEU score when decoding `test` corpus in REV preorder.

Decoding the REV preorder with $\lambda_{\text{CHILDBEFOREPARENT}} = -0.50$ (the weight that gave the highest BLEU score on the `dev` corpus) gave us our highest absolute `test` BLEU score, 28.74, among experiments conducted with $MaxDistortion = 9$. This represents a +0.78 increase over the comparable baseline, which is the BASELINE preorder decoded with $MaxDistortion = 9$ and only our PUNCT feature function.

5.9 Combining features

Our long-distance reordering features individually improved BLEU score. If we employ more than one at the same time, does the BLEU increase by the sum of the increase we saw for each feature on its own?

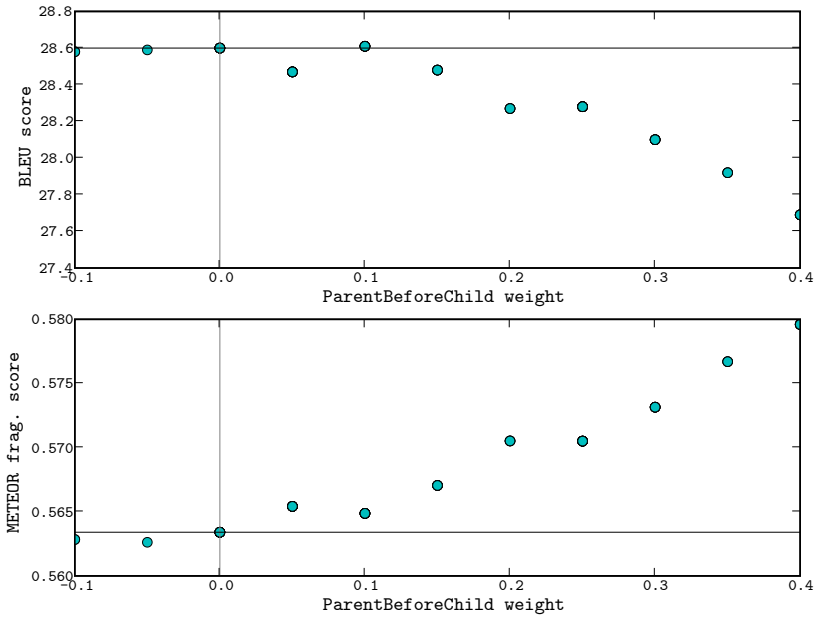

Figure 5-9: $\lambda_{\text{ParentBeforeChild}}$ against BLEU and METEOR fragmentation scores with Rev preorder.

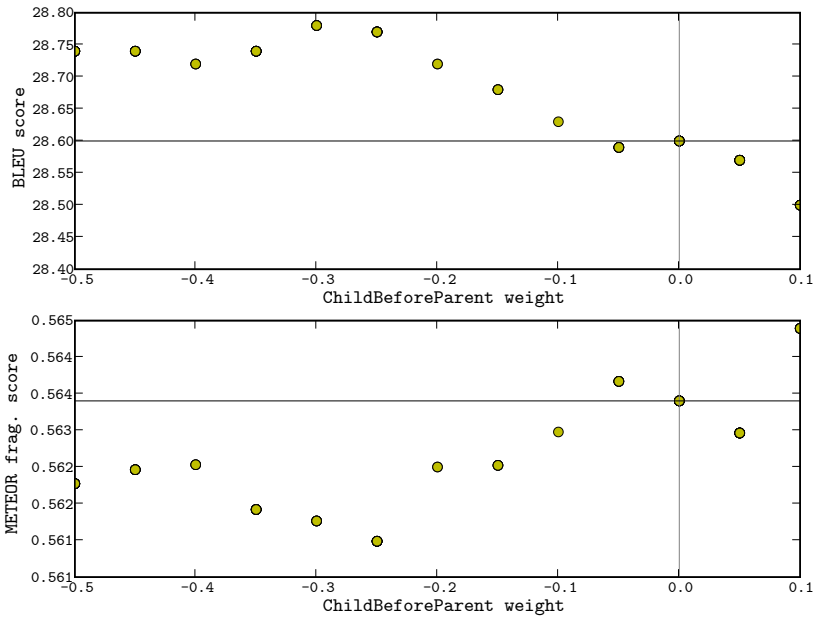

Figure 5-10: $\lambda_{\text{ChildBeforeParent}}$ against BLEU and METEOR fragmentation scores with Rev preorder.

First we look at the simple example of using ChildBeforeParent and ParentBeforeChild together. We found that `dev` BLEU score using both features did not improve over using just one of them, as evidenced by Table 5.8.

		$\lambda_{\text{ParentBeforeChild}}$	
		0	0.25
$\lambda_{\text{ChildBeforeParent}}$	0	26.99	27.12
	-0.30	27.08	27.09

Table 5.8: BLEU score on `dev` corpus when using ChildBeforeParent and ParentBeforeChild simultaneously.

To gauge performance of unisonal employment of VerbBeforeAcc, NounBeforeGen, and NounBeforeVerb, we found the weights of these features that gave highest scores individually on the `test` test corpus. Then we decoded the `test` corpus using all three features at the same time with those perfect weights.[7] The results are shown in Table 5.9. We see a total +0.58 increase using all three features. The maximum increase, if each feature contributed an increase equivalent to its standalone improvement, is +0.75. We can conclude ton the hat the features provide additive improvements in translation quality, but the improvement is less than the sum of the parts.

$\lambda_{\text{VerbBeforeAcc}}$	$\lambda_{\text{NounBeforeGen}}$	$\lambda_{\text{NounBeforeVerb}}$	`test` BLEU
0	0	0	27.96
0.10	0	0	28.06
0	0.40	0	28.45
0	0	0.30	28.12
0.10	0.40	0.30	28.54

Table 5.9: Performance of pairwise dependency features when combined.

5.10 Chunk cohesion

We introduced the ChunkCohesion feature in Section 3.6.2 to encourage chunks to be translated completely before moving on to translate other chunks. Figures 5-

[7] It would be proper experimental technique to report results based on weights tuned on the `dev`, here our aim is only to compare how effective features are alone versus combined.

11 and 5-12 show their effect on BLEU and METEOR fragmentation score when decoding preorder BASELINE and REV using a range of $\lambda_{\text{CHUNKCOHESION}}$.

The CHUNKCOHESION feature improved translation on the BASELINE preorder somewhat, but offered no improvement when translating the REV preorder. One plausible explanation is that the cohesion helps chunks move as a unit over long distances but is inutile for short movements.

Feature	Weight	`dev` BLEU	`test` BLEU
CHUNKCOHESION	-0.35	+0.21	+0.20

Table 5.10: Best scores for chunk cohesion feature.

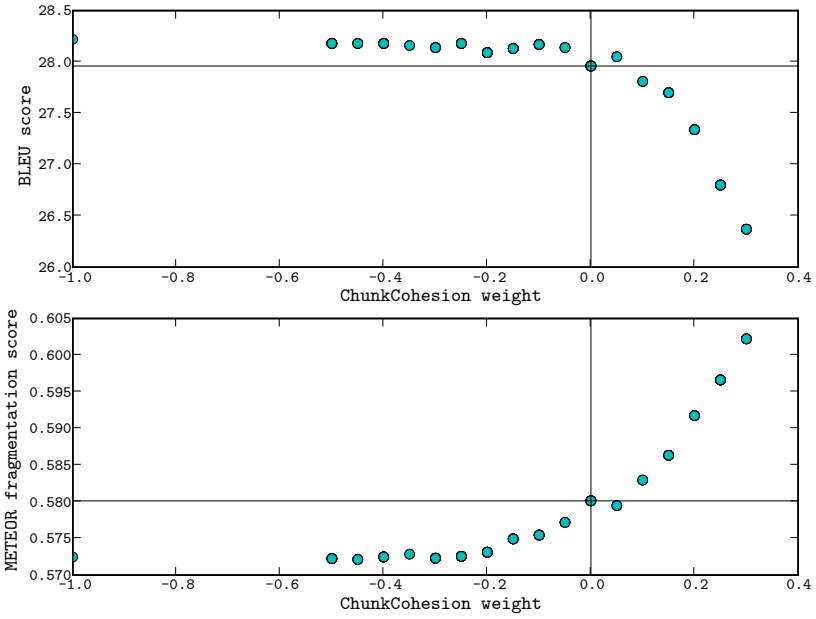

Figure 5-11: $\lambda_{\text{CHUNKCOHESION}}$ against BLEU and METEOR fragmentation scores with BASELINE preorder.

Unfortunately, when used with our other long-distance reordering features, CHUNKCOHESION does not increase translation quality. It offered maximum 0.02 BLEU score increase when used with feature weights $\lambda_{\text{VERBBEFOREACC}} = 0.15$, $\lambda_{\text{NOUNBEFOREGEN}} = 0.45$, and $\lambda_{\text{NOUNBEFOREVERB}} = 0.35$.

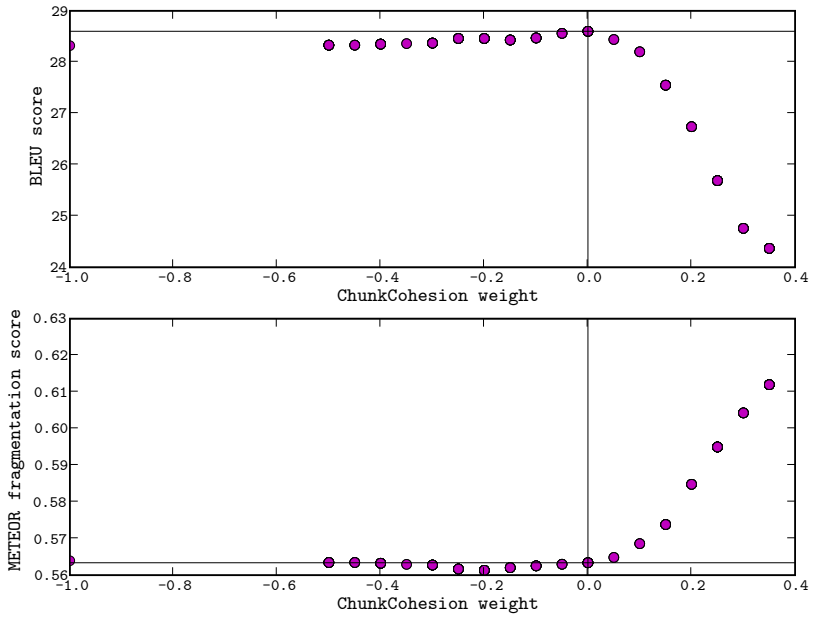

Figure 5-12: $\lambda_{\text{ChunkCohesion}}$ against BLEU and METEOR fragmentation scores with Rev preorder.

5.11 Punctuation

The Punct feature, introduced in Section 3.6.3 to discourage phrase movement over punctuation marks, did not prove especially helpful for BLEU score or subjective quality. With the Baseline preorder, $MaxDistortion = 9$, and weight $\lambda_{\text{Punct}} = -0.45$, this feature improved `dev` BLEU score by 0.20 and `test` by 0.10.

Subjectively, there was little systematic improvement to translations around punctuation. For the most part, we found that even without the Punct feature, phrases did not reorder over punctuation. Still, we found a few examples where this feature helped clausal cohesion around punctuation. Setting λ_{Punct} to a highly negative weight ensured that "for example" stayed inside the parenthetical in the following example.

Japanese	即ち、電気信号に変換され信号（例えば再生信号）として取り出される。
Reference	Specifically, the returned light L.sub.R is converted into an electrical signal (e.g., reproduced signal) and then output.
$\lambda_{\text{PUNCT}} = 0.05$	That is, the signal is converted into an electrical signal (a reproducing signal), for example.
$\lambda_{\text{PUNCT}} = -10.0$	In other words, the electric signal is converted to a signal (for example, a reproduced signal) is output.

Most of the time, however, word choice was randomly juggled in the vicinity of punctuation and not necessarily for the better. Here is one example where a more negative weight (λ_{PUNCT}) for PUNCT improved translation quality. The sequence "not limited to GaAs" is successfully translated.

Japanese	また、キャップ層６は、ＧａＡｓに限らず、オーミック接触のとりやすい物質、例えばＩｎＧａＡｓ等を用いてもよい。
Reference	Not only GaAs but also a material which makes the ohmic contact readily attainable, that is, InGaAs or the like, for example, may be used for the cap layer 6.
$\lambda_{\text{PUNCT}} = -0.05$	Further, the GaAs cap layer 6 is not limited to the material of the ohmic contact to tend to, for example, InGaAs or the like may be used.
$\lambda_{\text{PUNCT}} = -10.0$	Further, the cap layer 6 are not limited to GaAs, ohmic contact to tend to substance such as, for example, InGaAs or the like may be used.

Just as often, however, the translation jugglery is for the worse. A more strongly negative λ_{PUNCT} weight butchers this translation.

Japanese	この２分割デテクタは、２つのフォトダイオードＰＡ、ＰＢからなり、照射された反射レーザ光をそれぞれのフォトダイオードＰＡ、ＰＢにより検出する。
Reference	The two-segment detector 23 is made up of two photodiodes PA, PB for detecting the reflected laser light illuminated thereon.
$\lambda_{\text{PUNCT}} = -0.05$	The two-division detector includes two photodiode Pa, PB, and the irradiation of the reflected laser light are detected by the photodiode Pa and Pb.
$\lambda_{\text{PUNCT}} = -10.0$	The two-division detector includes two photodiode Pa, PB, and the reflected laser beam is irradiated onto the photo diode Pa, PB, the detected.

5.12 Minimum error rate training

Moses comes with a script to perform Minimum Error Rate Training (MERT) to tune feature weights to maximize BLEU score on a development corpus [Och and Ney, 2001; Koehn *et al.*, 2007]. However, preliminary experiments showed that running MERT training, adding a new feature, then rerunning MERT training often resulted in lower `dev` scores. Some score randomness is expected because the Moses MERT algorithm is not deterministic, but we decided that for evaluating the effect of new features on scores, it was best to tune parameters by hand. One reason for the poor results using the Moses MERT script may be that it was designed and tested with the small number of default Moses features, which number 10 to 20 depending on configuration. Liang *et al.* [2006] successfully developed a discriminatively trained system with millions of features. Using their parameter tuning method would be effective to tune weights for our features, and would open the door to adding features to model many more part of speech and case dependency relations.

The weights learned from using the Moses MERT script, presented in Table 5.11, hint at the promise of the MERT technique.[8] Each feature is automatically given a

[8] The weights in Table 5.11 cannot be compared with weights in other tables, because they have been normalized alongside the baseline Moses weights.

weight that pushes translations toward correct English word order; that is, the MERT tuning correctly identifies whether to scale each feature positively or negatively to improve `dev` translations.

Feature	Weight
PUNCT	-0.015772
CHUNKCOHESION	-0.004921
PARENTBEFORECHILD	0.017132
CHILDBEFOREPARENT	-0.219797
VERBBEFOREACC	0.126323
NOUNBEFOREGEN	0.102700
NOUNBEFOREVERB	0.098392

Table 5.11: Feature weights after minimum error rate training.

The baseline Moses distortion penalty weight (which penalizes non-monotonic translations, see Section 3.4.4) was also noteworthy at -0.000135. The negative value indicates that, when using our new feature functions, the decoder could achieve better translations by preferring non-monotonic translations. In contrast, MERT set the distortion penalty weight to a value greater than 0.01 every time we tuned parameters on a system that did not include our long-distance reordering features.

5.13 BLEU versus METEOR for evaluating word order quality

When scoring a hypothesis translation against a reference, BLEU focuses only on how many n-grams in the hypothesis match the reference, and otherwise ignores word order completely. Because BLEU typically counts up to 4-grams, it does not explicitly factor long-distance word order into the score at all. Callison-Burch *et al.* [2006] note that if b is the number of bigram mismatches (pairs of words that appear together in the hypothesis translation but not the reference), then there are $(k-b)!$ possible ways, to generate identically scored translations using only the words in the hypothesis. Hence theoretically BLEU seems unable to distinguish differences in word order between translation systems. METEOR in contrast explicitly incorporates a fragmentation

score, which measures how dissimilar the word order is among words that appear in both. The METEOR metric makes the assumption that the lower the fragmentation score, the better the word order.

In the plots in this chapter, we compare BLEU score and METEOR fragmentation score on our experiments where we range the weight of one long-distance reordering function while keeping all other system parameters the same. For a feature that clearly should have a positive effect on English word order, like NounBeforeGen, we expected the METEOR fragmentation score to have a positive slope around zero until a peak in translation quality. We expected BLEU score to increase, but not as systematically.

If anything, the plots show the opposite phenomenon: BLEU score had a systematic positive slope as the beneficial feature weight increased, while METEOR fragmentation score tended to bounce around. The plots for $\lambda_{\text{VerbBeforeAcc}}$, $\lambda_{\text{NounBeforeGen}}$, and $\lambda_{\text{NounBeforeVerb}}$ in Figures 5-3–5-5 are an interesting sample to look at. For $\lambda_{\text{VerbBeforeAcc}}$, BLEU score is better than the baseline for all $\lambda_{\text{VerbBeforeAcc}} \leq 0.45$, which indicates that translation quality is increasing. Meanwhile METEOR fragmentation score is higher than the baseline for all values of $\lambda_{\text{VerbBeforeAcc}}$ except 0.15, which indicates that translate quality is decreasing, at least word order wise. Looking subjectively at the translations, it more sentences are improved word-order wise than are harmed.

For $\lambda_{\text{NounBeforeVerb}}$ (Figure 5-5), another subjectively beneficial feature, BLEU monotonically increases from weight 0 to 0.3, while METEOR score is scattered and reaches its highest value (indicating worst quality) at weight 0.3. Either BLEU or METEOR fragmentation score is making a mistake, and the evidence that our features *do* improve on word order leads us to conclude that BLEU is capturing word order differences in translations better than METEOR fragmentation score. This is not to say that METEOR is a bad metric; this is merely evidence that its fragmentation component is likely not a great indicator of word order quality. It is also clear that BLEU is not a fantastic metric for evaluating changes in system word order; we interpret the plots and our subjective judgments merely as evidence that

BLEU is not totally useless for evaluating word order choices in translation. This is likely because translations with words in proper order simply generate more n-gram matches with the reference.

Chapter 6

Conclusion

This thesis developed two techniques to improve long-distance reordering decisions in the phrase-based translation model and demonstrated their utility in a state-of-the-art Japanese→English system. Chapter 3 introduced our major contribution, a set of long-distance reordering feature functions that use a dependency analysis of the source sentence to encourage translations that reorder phrases in a way that preserves their original meaning. Chapter 4 presented algorithms for reordering Japanese into an English word order before translation, with the surprising result that a naive preprocessor that basically flips the Japanese to read backwards outperforms a dependency-tree flattening method we developed. Experiments in Chapter 5 demonstrated significant improvement in BLEU score and subjective quality in experiments with both methods and further gains when we combined them.

6.1 Future work

Current statistical translation systems have a long way to go to achieve perfect word order for languages requiring long-distance reordering. Our pairwise dependency order features are only the beginning of what is possible when incorporating dependency analysis into phrase-based models.

6.1.1 Smarter reordering limit

Translation quality increses when we allow unlimited reordering of phrass, but translation speed becomes prohibitively slow. Current phrase-based systems offer little recourse if we wish to limit reordering but still consider linguistically-motivated long-distance reordering. The ubiquitous *MaxDistortion* limit is a vestige of systems that favor monotone translation and causes quality hemorrhage in language pairs that require long-distance reordering.

A discriminatively trained model for limiting reordering based on a dependency tree distance metric could help the decoder to speedily try all of the important long-distance reorderings. For instance, after the decoder completely translates the subject of a sentence into English, the distortion limit should force the decoder to next translate a phrase that is within a certain distance from the verb that the subject modifies. The challenge is training a discriminative order model that is part of the decoder's internal machinery.

6.1.2 More effective features

It is critical to identify translation hypotheses with promising word order as early as possible to avoid the decoder pruning them. For example, it is undesirable that the pairwise dependency order features of Section 3.6.1 have value zero until both the child and parent have been translated. We should experiment with features that have nonzero contribution as soon as either the child or the parent is translated, because at that point we can infer that the other member of the dependency relationship will be translated after it, based on the assumption that the decoder always builds its translation left–to–right. This would allow earlier detection, and less pruning, of correct word orders. To concretize this idea, an improved version of $\chi_{\textsc{ParentBeforeChildTemplate}}(f_2^J, q, s)$ is given in Equation 6.1. This version contributes a nonzero value to a hypothesis as soon as the parent chunk is translated.

$$\chi_{\textsc{ImprovedParentBeforeChildTemplate}}(f_1^J, q, s)$$
$$= \sum_{x \in X} \left\{ \text{matches\_parent}(x, s) \cdot \sum_{y \in x.\text{children}} \left\{ 1 - \left(\begin{array}{l} \text{matches\_child}(y, s) \\ \cdot \text{frac\_already\_covered}(y, q) \\ \cdot \text{frac\_translated}(x, q) \end{array} \right) \right\} \right\} \tag{6.1}$$

If our dependency-based features were integrated into a discrminative training system with support for millions of features, we could introduce features that are are parameterized on head and modifier words themselves in addition to their parts of speech. Features that measure how far modifiers move away from their head or the order of dependency tree siblings may also improve translation quality.

6.1.3 Other language pairs

Because our long-distance reordering features make no assumptions about source or target language word order, they should be easily appliable to any phrase-based system. Experimenting on other language pairs is an extremely exciting prospect. In particular, features promoting verbal head movement should be very useful for English→Japanese translation to help verbs to reorder to the right of all of their modifiers.

6.2 Contributions

To translate between Japanese and English, or any language pair with very different word order, we need a translation system that can perform long-distance reordering while preserving the meaning of the original input. Towards this goal, this thesis:

- Designed a class of feature functions for phrase-based translation that can identify translations with correct long-distance reordering for any language pair.

- Implemented these features in a state-of-the-art phrase-based decoder to achieve significant improvements in Japanese→English BLEU score and subjective translation quality.
- Remedied to a significant extent the problem of leaving Japanese verbs sentence-final and genitive constructions inverted when translating into English, which plagues most statistical phrase-based translation systems.
- Demonstrated a naive, trivially computable source-side preordering algorithm that dramatically increases Japanese→English translation quality when decoding with limited allowed reordering.
- Provided evidence that BLEU is useful for evaluating quality of translations that differ mostly in word order.

Bibliography

[ACL, 2005] The Association for Computer Linguistics. *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, 2005.

[ACL, 2006] The Association for Computer Linguistics. *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*, 2006.

[ACL, 2007] The Association for Computer Linguistics. *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*, 2007.

[Al-Onaizan and Papineni, 2006] Yaser Al-Onaizan and Kishore Papineni. Distortion models for statistical machine translation. In ACL [2006].

[Breen, 1995] Jim Breen. Building an electronic Japanese-English dictionary. In *Proceedings of 1995 Japanese Studies Association of Australia Conf.*, 1995.

[Brown *et al.*, 1993] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311, 1993.

[Callison-Burch *et al.*, 2006] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of the 11th Conference of the European Chapter of the Association of Computational Linguistics*, pages 249–256, 2006.

[Chang and Toutanova, 2007] Pi-Chuan Chang and Kristina Toutanova. A discriminative syntactic word order model for machine translation. In ACL [2007].

[Cherry, 2008] Colin Cherry. Cohesive phrase-based decoding for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 72–80, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[Chiang, 2007] David Chiang. Hierarchical phrase-based translation. Computational Linguistics 33(2):201-228, 2007.

[Collins *et al.*, 2005] Michael Collins, Philipp Koehn, and Ivona Kucerova. Clause restructuring for statistical machine translation. In ACL [2005].

[Cowan *et al.*, 2006] Brooke Cowan, Ivona Kučerová, and Michael Collins. A discriminative model for tree-to-tree translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 232–241, Sydney, Australia, July 2006. Association for Computational Linguistics.

[Cross Language, 2008] Cross Language company information. `http://www.crosslanguage.co.jp/company/`, August 2008.

[Crystal, 1997] David Crystal. *The Cambridge Encyclopedia of Language*. Cambridge University Press, 2 edition, 1997.

[Ding and Palmer, 2005] Yuan Ding and Martha Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In ACL [2005].

[Dyer, 2007] Christopher J. Dyer. The "noisier channel": Translation from morphologically complex languages. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 207–211, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Fujii *et al.*, 2007] Atsushi Fujii, Takehito Utsuro, Mikio Yamamoto, and Masao Utiyama. Description of patent translation task at NTCIR-7. `http://if-lab.slis.tsukuba.ac.jp/fujii/ntc7patmt/task.pdf`, 2007.

[Huang *et al.*, 2006] Liang Huang, Kevin Knight, and Aravind Joshi. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*, Boston, MA, 2006.

[Kanthak *et al.*, 2005] Stephan Kanthak, David Vilar, Evgeny Matusov, Richard Zens, and Hermann Ney. Novel reordering approaches in phrase-based statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pages 167–174, 2005.

[Knight, 1999] Kevin Knight. A statistical MT tutorial workbook. Prepared in connection with the JHU summer workshop, April 1999.

[Koehn *et al.*, 2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Koehn *et al.*, 2005] Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh system description for the 2005 IWSLT Speech Translation Evaluation. In *International Workshop on Spoken Language Translation 2005*, 2005.

[Koehn *et al.*, 2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Koehn, 2003] Philipp Koehn. *Noun Phrase Translation*. PhD thesis, University of Southern California, 2003.

[Koehn, 2007] Philipp Koehn. Moses background. `http://www.statmt.org/moses/?n=Moses.Background`, 2007. Explanation of Moses's approach excerpted from [Koehn, 2003].

[Kudo and Matsumoto, 2002] Taku Kudo and Yuji Matsumoto. Japanese dependency analysis using cascaded chunking. In *CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops)*, pages 63–69, 2002.

[Kudo, 2007] Taku Kudo. Mecab: Yet another part-of-speech and morphological analyzer, 2007.

[Kuhn *et al.*, 2006] Roland Kuhn, Denis Yuen, Michel Simard, Patrick Paul, George F. Foster, Eric Joanis, and Howard Johnson. Segment choice models: Feature-rich models for global distortion in statistical machine translation. In Robert C. Moore, Jeff A. Bilmes, Jennifer Chu-Carroll, and Mark Sanderson, editors, *HLT-NAACL*. The Association for Computational Linguistics, 2006.

[Lavie and Agarwal, 2007] Alon Lavie and Abhaya Agarwal. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Li *et al.*, 2007] Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li, Ming Zhou, and Yi Guan. A probabilistic approach to syntax-based reordering for statistical machine translation. In ACL [2007].

[Liang *et al.*, 2006] Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Benjamin Taskar. An end-to-end discriminative approach to machine translation. In ACL [2006].

[Lin, 2004] Dekang Lin. A path-based transfer model for machine translation. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 625, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[Liu *et al.*, 2006] Yang Liu, Qun Liu, and Shouxun Lin. Tree-to-string alignment template for statistical machine translation. In ACL [2006].

[Menezes and Quirk, 2007] Arul Menezes and Chris Quirk. Using dependency order templates to improve generality in translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 1–8, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Menezes *et al.*, 2006] Arul Menezes, Kristina Toutanova, and Chris Quirk. Microsoft research treelet translation system: NAACL 2006 europarl evaluation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 158–161, New York City, June 2006. Association for Computational Linguistics.

[Nichols *et al.*, 2007] Eric Nichols, Francis Bond, Darren Scott Appling, and Yuji Matsumoto. Combining resources for open source machine translation. In *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 134–143, Skövde, Sweden, 2007.

[NIST, 2006] National Institute of Standards and Technology (NIST) 2006 Machine Translation Evaluation official results. `http://www.nist.gov/speech/tests/mt/2006/doc/mt06eval_official_results.html`, November 2006.

[Och and Ney, 2001] Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[Och and Ney, 2004] Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449, 2004.

[Och *et al.*, 2004] Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alexander Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir R. Radev. A smorgasbord of features for statistical machine translation. In *HLT-NAACL*, pages 161–168, 2004.

[Papineni *et al.*, 2001] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[Quirk *et al.*, 2005] Christopher Quirk, Arul Menezes, and Colin Cherry. Dependency treelet translation: Syntactically informed phrasal SMT. In ACL [2005].

[Riezler and Maxwell III, 2006] Stefan Riezler and John T. Maxwell III. Grammatical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 248–255, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[Suzuki and Toutanova, 2006] Hisami Suzuki and Kristina Toutanova. Learning to predict case markers in Japanese. In ACL [2006].

[Tanaka, 2001] Yasuhito Tanaka. Compilation of a multilingual parallel corpus. In *Proceedings of PACLING 2001*, Fukuoka, Japan, 2001.

[Tillmann and Zhang, 2005] Christoph Tillmann and Tong Zhang. A localized prediction model for statistical machine translation. In ACL [2005].

[Tillmann, 2004] Christoph Tillmann. A unigram orientation model for statistical machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Short Papers*, pages 101–104, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[Utiyama and Isahara, 2003] Masao Utiyama and Hitoshi Isahara. Reliable measures for aligning Japanese-English news articles and sentences. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 72–79, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Utiyama *et al.*, 2007] Masao Utiyama, Mikio Yamamoto, Atsushi Fujii, and Takehito Utsuro. Description of patent parallel corpus for NTCIR-7 patent translation task. `http://if-lab.slis.tsukuba.ac.jp/fujii/ntc7patmt/ppc.pdf`, 2007.

[Wang *et al.*, 2007] Chao Wang, Michael Collins, and Philipp Koehn. Chinese syntactic reordering for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 737–745, 2007.

[WMT Baseline, 2007] ACL 2007 Second Workshop on Statistical Machine Translation shared task: Baseline system. `http://www.statmt.org/wmt07/baseline.html`, 2007.

[Xiong *et al.*, 2006] Deyi Xiong, Qun Liu, and Shouxun Lin. Maximum entropy based phrase reordering model for statistical machine translation. In ACL [2006].

[Xiong *et al.*, 2007] Deyi Xiong, Qun Liu, and Shouxun Lin. A dependency treelet string correspondence model for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 40–47, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Yahoo, 2008] Yahoo! Babel Fish. `http://babelfish.yahoo.com`, August 2008.

[Zens *et al.*, 2004] Richard Zens, Hermann Ney, Taro Watanabe, and Eiichiro Sumita. Reordering constraints for phrase-based statistical machine translation. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 205, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[Zhang *et al.*, 2007] Dongdong Zhang, Mu Li, Chi-Ho Li, and Ming Zhou. Phrase reordering model integrating syntactic knowledge for SMT. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 533–540, 2007.