Statistical Machine Translation Models Utilizing Tree Combination

Zhongyuan Zhu

(Master's Program in Computer Science)

Advised by Mikio Yamamoto

Submitted to the Graduate School of
Systems and Information Engineering
in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering
at the
University of Tsukuba

March 2014

Abstract

In this thesis, we propose a method to combine CFG and dependency parse trees, and utilize the combined trees as the input of translation model. The combined parse trees keep head-dependent relationships in large span and reasonable structures for local phrases, which benefit the translation for language pairs with sharp contrast in word order.

Our experiments show improved translation accuracy using the proposed model when compared to that obtained by translating CFG parse trees alone, especially for translating long sentences. The official human evaluations operated in the Patent Machine Translation Task of NTCIR-10 Workshop show our translation model produces more adequate results comparing with the state-of-the-art Hierarchical phrase-based translation model.

Contents

List of Figures

Chapter 1

Introduction

1.1 Machine translation

The goal of machine translation is simple: to translate sentences or documents from one natural language to another in an automatic fashion. However, this does not mean it is easy to achieve. Since the digital computer was developed, attempts to translate using computers have never stopped. However, even today, state-of-the-art machine translation systems do not always generate acceptable results. For example, when we translate "We will miss John" using Google's Translation Service, it generates the result "我々はジョンを欠場する" (this trial was performed on Jan. 12, 2014). In addition, it is not unusual for these systems to produce ungrammatical or incomprehensible translations.

In recent years, many commercial and free online translation systems have been developed. Most of them are built on the basis of Rule-Based Machine Translation (RBMT) technology that utilize linguistic knowledge of the relationships between the source and target language. Basically, this kind of knowledge is retrieved from bilingual dictionaries and other resources and built into the translation systems manually. Large amounts of morphological, syntactical, and even semantic knowledge are required so that RBMT systems can produce high-quality translations. Hence, the cost of developing these systems is very high.

Statistical Machine Translation (SMT) allows translation systems to be built directly on the bilingual corpora of translated text without additional linguistic knowledge. SMT systems automatically discover and learn translation rules, and assign probabilities to them. As the amount of Internet content increases, large amounts of bilingual translated text have become easier to collect. Based on this, SMT systems are expected to be applied to more language pairs. Compared to RBMT systems, SMT systems are also much easier to develop and maintain; even a Master's student can build one from scratch.

Although SMT systems have already achieved high performance in some major language pairs such as French-English and Arabic-English, some problems still need to be tackled for translations between languages that significantly differ in word order and syntax, such as Japanese-English.

1.2 About this thesis

To produce better translations for language pairs such as English and Japanese, syntax parse trees are generally applied to guide the translation process. CFG parse trees provide syntax structures for sentences, but as English and Japanese hugely differ in the syntax, translation models using CFG parse trees alone can hardly capture long-range translation relationships. By contrasy, dependency parse trees provide head-dependent relationships of the words in a whole sentence, but the reasonable local phrase structures are lost.

In this thesis, we attempt to capture the long-range correspondences of translation pairs in a discriminating way. We combine CFG and Dependency parse trees as the input of our translation model. The combined tree keeps both the structural information and dependency relationships, which is helpful for long sentence translations.

In our evaluations, we found our model could generally achieve better automatic evaluation scores compared to our baseline models that translate CFG parse trees alone. For long sentences (above 40 words), our model outperforms the state-of-the-art Hierarchical phrase model in an English-Japanese translating task. In the NTCIR-10 Workshop (Goto et al., 2012), our model outperformed two official baseline models (Hierarchical phrase and Phrase-based models) in human evaluations.

In the rest of this thesis, we first introduce CFG and Dependency parse trees and briefly review Statistical Machine Translation in Chapter 2. We then present the tree-to-string SMT approach in Chapter 3. We describe our methods for combining CFG and Dependency parse trees, our rule acquisition, the feature set of our model and the decoding process in Chapter 4. Evaluations are presented in Chapter 5.

Chapter 2

Preliminaries

2.1 Context free grammars

Context free grammars (CFG) (Chomsky & Noam, 1956) are important and widely used in the researches of linguistics and Nature language processing, which describes the structure a sentence. Traditionally, a context free grammar is defined as (N, Σ, R, S). Where in the tuple, N is a finite set of non-terminals. Σ is a finite set of terminals. R is a set of production rules, a rule (X, α) where $X \in N, \alpha \in (\Sigma \cup N)^*$ is usually presented in the form $X \rightarrow \alpha$. The last symbol $S \in N$ is a start symbol. Figure 2.1 shows an example of derivation in the form of tree presentation, where $A, B, C \in N; a, b, c \in \Sigma$ and the arrows show the generating relations of derivation.

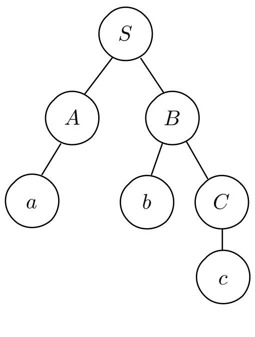


Figure 2.1: An example derivation of CFG

This derivation can be generated by following production rules:

$$\begin{array}{lcl} S & \rightarrow & A\ B \\ A & \rightarrow & a \\ B & \rightarrow & b\ C \\ C & \rightarrow & c \end{array}$$

2.2 Dependency grammar

In most papers, the origin of modern dependency grammar is regarded to be French linguist Lucien Tesni'ere (1959), though some researches argue that dependency grammar is already been used from the Middle Ages. The intuition of dependency grammar is to describe words in a sentence not by their structures, but dependency relationships. For the example "John hit a ball", we can demonstrate these relationships in flowing way:

1) *John* is the subject of the verb *hit*
2) *ball* is the object of the verb *hit*
3) *a* is an article that modifies the noun *ball*

Figure 2.2 shows an illustration of the dependency relationship of this example. Where "nsubj", "dobj" and "det" indicates the types of dependency relationships for each modifier.

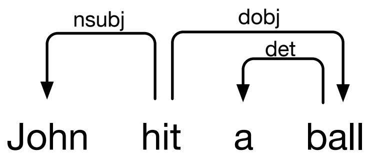


Figure 2.2: An illustration of dependency relationship

2.3 Statistical machine translation

The approach of Statistical Machine Translation was initiated by Brown et al.(1990) . In this section, we will introduce some most influential models in this approach, and a common method for tuning translation models. At the end of this chapter we will explore some methods for evaluating translation results.

2.3.1 Word based models

Around 1990, as the initial work of IBM group, a series of so called IBM models (IBM model 1 to 5) were proposed (Brown et al., 1993) , which formed the foundation of Statistical Machine Translation today. Given a sentence $f_1^I = f_1...f_I$ with I words in source language which is to be translated to target sentence $e_1^J = e_1...e_J$. Where f means French and e stands for English in papers of IBM models. Intuitively, if a word f_i in source language and another word e_j in target language always appear simultaneously in bilingual corpus, then e_j is more likely to be the translation of f_i. IBM Model 1 gives the formula to calculate translation probabilities as follows, which ignores the factor of lexical reordering. Where **a** in the formula indicates alignment pairs, a set of correspondences between words in source and target side.

$$P(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

For higher IBM Models, more features are considered. In IBM Model 2, absolute alignment probability distribution is combined. Then IBM Model 3 adds fertility model. IBM Model 4 improves IBM Model 3, which introduces a relative distortion model. IBM Model 5 eliminates the problem of deficiency in IBM Model 3 and 4.

Although many newer translation models in subsequent improved translation quality comparing with IBM Models. They are still state-of-the-art in training word alignments, which implemented in a widely-used tool, GIZA++(Och & Ney, 2004).

2.3.2 Log-linear modeling

As a basic framework of SMT systems, Noisy-channel model is highly influential for current SMT systems. By applying Bayes Rule, the best translation hypothesis $\hat{e}_1^J$ can be obtained by following formula

$$\begin{aligned}
\hat{e}_1^J &= \operatorname*{argmax}_{e_1^J}\{p(e_1^J|f_1^I)\} \\
&= \operatorname*{argmax}_{e_1^J}\{\frac{p(f_1^I|e_1^J)p(e_1^J)}{p(f_1^I)}\} \\
&= \operatorname*{argmax}_{e_1^J}\{p(f_1^I|e_1^J)p(e_1^J)\}.
\end{aligned}$$

In this model, the posterior probability is decomposed into two probability distributions. $p(f_1^I|e_1^J)$ is commonly recognized as so-called the translation model, which gives the confidence of e_1^J to be a correct translation of f_1^I. $p(e_1^J)$ is so-called the language model which evaluates the acceptability of e_1^J in target language. Figure 2.3 shows the architecture of SMT systems implemented with Noisy-channel model. During the translation process, the decoder search for the best translation hypothesis according to two models above.

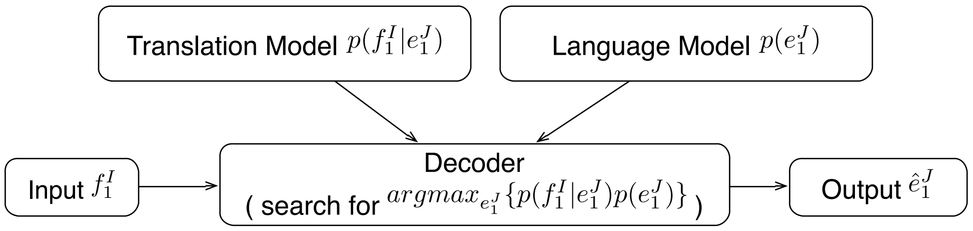

Translation Model $p(f_1^I|e_1^J)$
Language Model $p(e_1^J)$
Input f_1^I
Decoder
(search for $argmax_{e_1^J}\{p(f_1^I|e_1^J)p(e_1^J)\}$)
Output $\hat{e}_1^J$

Figure 2.3: Architecture of SMT systems with Noisy-channel model

However, the state-of-the-art SMT systems today apply Log-linear model (Och & Ney, 2002) instead of Noisy-channel model. Log-linear model decomposes translation probability

$p(e_1^J|f_1^I)$ into several feature functions. Each feature function is assigned with a weight factor. In this model, the best translation hypothesis can be obtained in following formula

$$\begin{aligned}
\hat{e}_1^J &= \underset{e_1^J}{\operatorname{argmax}}\{p(e_1^J|f_1^I)\} \\
&= \underset{e_1^J}{\operatorname{argmax}}\{\frac{\exp(\sum_{m=1}^{M}\lambda_m h_m(f_1^I, e_1^J))}{\sum_{e'^J_1}\exp(\sum_{m=1}^{M}\lambda_m h_m(f_1^I, e'^J_1))}\} \\
&= \underset{e_1^J}{\operatorname{argmax}}\{\exp(\sum_{m=1}^{M}\lambda_m h_m(f_1^I, e_1^J))\} \\
&= \underset{e_1^J}{\operatorname{argmax}}\{\sum_{m=1}^{M}\lambda_m h_m(f_1^I, e_1^J)\}.
\end{aligned}$$

Log-linear model provides more flexibility for adding extra knowledge-based features into SMT system, and Noisy-channel model can be considered as a special case of Log-linear model. Weight factor λ_m of each feature function is usually tuned with BLEU score on an independent development data set.

2.3.3 Phrase-based machine translation

In IBM models, only lexical probabilities are considered, which focus on modelling the probabilities of translations for a single word. But other contextual information such as co-occurrence of multiple words is not taken into account. Phrase-based translation model proposed by Och et al. (Och & Tillmann+, 1999; Zens & Och+, 2002; Koehn & Och+, 2003) is a successor of word-based models, which models the probabilities of translating phrases. In this model and following models, the terminology "phrase" does not have linguistic meaning, but just indicates an arbitrary sequence of words.

Phrase-based model breaks up input sentence $\bar{f}_1^I$ into several phrases $\bar{f}_1 ... \bar{f}_I$, then translates each phrase $\bar{f}_i$ into $\bar{e}_i$ respectively. Finally, translated phrases $\bar{e}_1 ... \bar{e}_I$ are reordered into appropriate positions. In the mathematical definition of Phrase-based model, $p(\mathbf{f}|\mathbf{e})$ is decomposed into phrase translation probability and distortion probability, as following formula shows

$$p(\mathbf{f}|\mathbf{e}) = p(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I}\phi(\bar{f}_i|\bar{e}_i)d(a_i - b_{i-1}).$$

In this formula, a_i indicates the start position of $\bar{f}_i$, b_{i-1} indicates the ending position of the source phrase of the left target phrase of $\bar{e}_i$. A exponentially penalty function with a parameter α is applied for distortion probability as following equation shows

$$d(a_i - b_{i-1}) = \alpha^{|a_i - b_{i-1} - 1|}.$$

Phrase translation probability $\phi(\bar{f}_i|\bar{e}_i)$ is computed based on relative frequency

$$\phi(\bar{f}_i|\bar{e}_i) = \frac{\text{Count}(\bar{f}, \bar{e})}{\sum_{\bar{f}'} \text{Count}(\bar{f}', \bar{e})}.$$

Rule acquisition

As the output of IBM models, word alignments with the optimal probability are generated. An example of word alignments in English-to-Japanese translation task is shown in Figure 2.4. It could also be presented in the form of lattice, as the right side of Figure 2.4.

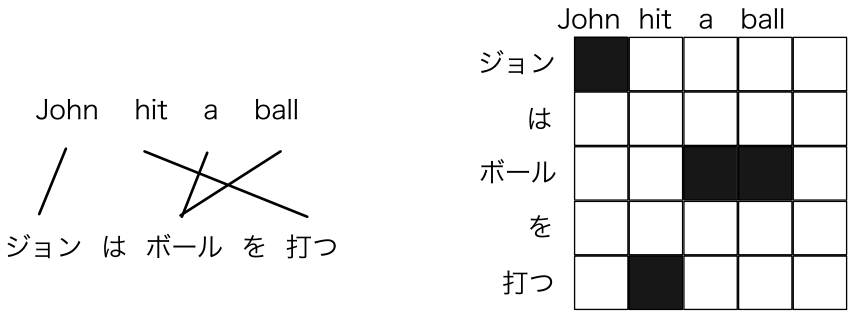


Figure 2.4: An example of word alignments

For current state-of-the-art SMT systems, translation rules are automatically acquired using this kind of word alignments. Phrase-based model extracts a phrase pair $(\bar{f}_i, \bar{e}_i)$ if it is consistent with given word alignments A. Here is the precise definition of consistency:

$(\bar{f}_i, \bar{e}_i)$ is consistent with A iff:

1) $\forall e_i \in \bar{e} : (f_i, e_i) \in A \Rightarrow f_i \in \bar{f}$
2) $\forall f_j \in \bar{f} : (f_i, e_i) \in A \Rightarrow e_i \in \bar{e}$
3) $\exists e_i \in \bar{e}, f_i \in \bar{f} : (f_j, e_i) \in A$

These three requirements assure that for any aligned word in an extracted phrase pair, its source or target word is also included in this phrase pair. Unaligned words are also allowed to exist in an extracted phrase pair, but those phrases only composed with unaligned words are excluded. Some possible examples of extracted phrase pairs are shown in Figure 2.5.

Model

Phrase-based model applies 6 features for each phrase pair to form a translation rule, hence they could be combined into Log-linear model:

1) Phrase translation probability $\log(\phi(\bar{e}|\bar{f}))$ from source to target

2) Phrase translation probability $\log(\phi(\bar{f}|\bar{e}))$ from target to source

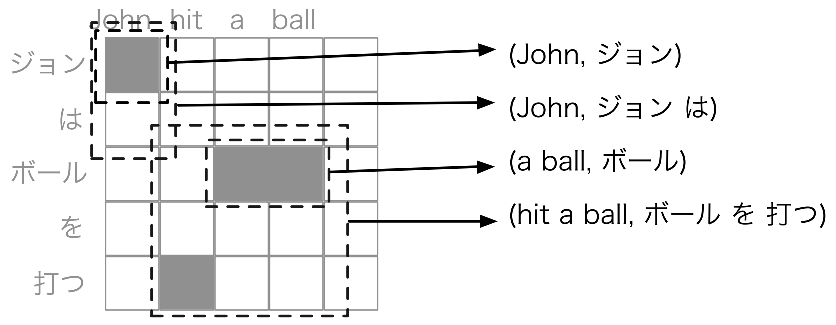


Figure 2.5: Some possible examples of extracted phrase pairs

3) Lexical translation probability $\log(\text{lex}(\bar{e}|\bar{f}, a))$ from source to target

4) Lexical translation probability $\log(\text{lex}(\bar{f}|\bar{e}, a))$ from target to source

5) Phrase penalty: 1

6) Word penalty: $\text{Count}(\bar{e})$

For lexical translation probability, many different methods are proposed. A simple one is based on the word translation probability of IBM model 1:

$$lex(\bar{e}|\bar{f}, \mathbf{a}) = \prod_{i=1}^{length(\bar{e})} \frac{1}{|\{j|(i,j) \in \mathbf{a}\}|} \sum_{\forall(i,j)\in a} w(e_i|f_j)$$

Decoding

In decoding phase, the decoder searches for the best translation hypothesis by given input sentence. As Figure 2.6 illustrates, firstly, an empty initial hypothesis is made. Then begin with this hypothesis, a phrase with arbitrary length is selected in each iteration, translated phrase is added into the end of sentence in target side. The decoding process ends if all words of input sentence are translated, the best hypothesis with highest score is selected to be the final translation result.

If a long sentence is inputted, the total number of possible hypothesis will increase exponentially. So in most implementations, a technique called Beam Search (Koehn & Philipp, 2004) is normally applied, which forces the decoder to keep only k-best hypotheses in the hypothesis stack during decoding.

2.3.4 Hierarchical Phrase-based machine translation

In previous section, we presented Phrase-based translation model, which considers the adjacent context in the translation process. However, Phrase-based model restricts the words contained in a phrase must be consecutive, which makes the translation unable to capture

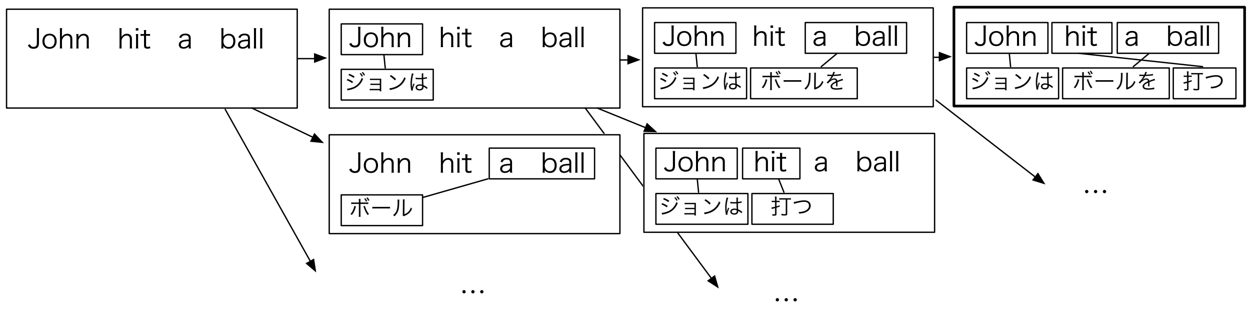
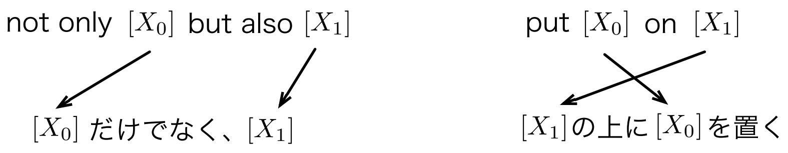


Figure 2.6: Illustration of decoding process of Phrase-based model

long-distance translation pairs with gaps involved. Two common examples of translation pairs with gaps are shown in Figure 2.7. Where $[X_0]$ and $[X_1]$ in the phrases indicate gaps that can be replaced with any other phrase pair.



not only $[X_0]$ but also $[X_1]$
$[X_0]$ だけでなく、$[X_1]$
put $[X_0]$ on $[X_1]$
$[X_1]$の上に$[X_0]$を置く

Figure 2.7: Two examples of translation pair with gaps

To capture this kind of long range dependencies of words, Hierarchical Phrase-based translation model was proposed by Chiang (Chiang, 2005), which is one of the most influential translation models currently.

Synchronous CFG

Hierarchical Phrase-based translation model, and also proposed translation models described in following chapters are based on Synchronous CFG (SCFG) (Aho and Ullman, 1969), which modeling the reordering of sub-phrases both in source and target side of a translation pair. It has the form:

$$X \rightarrow < \gamma, \alpha, \sim >$$

In this expression, X is a nonterminal, γ and α are strings mixed with nonterminals and terminals, $\sim$ indicates correspondence relationships between nonterminals in both side. For example, the translation pairs shown in Figure 2.7 can be simply presented in following forms:

$$X \rightarrow < \text{not only } X_0 \text{ but also } X_1, \ X_0 \text{ だけでなく、 } X_1 >$$
$$X \rightarrow < \text{put } X_0 \text{ on } X_1, \ X_1 \text{ の上に } X_0 \text{ を置く } >$$

Rule Acquisition

As what we described in the section of Phrase-based model, the extraction process of translation rules begins with automatized learned word alignments of IBM models. First, a set of initial phrases are extracted in the same manner of Phrase-based model. Second, in order to obtain hierarchical translation rules, all phrases that contain initial phrases are observed. To form a translation rule, those initial phrases are replaced with a nonterminal symbol X_n to indicate a gap between words. An illustration of extracting hierarchical phrases is shown in Figure 2.8. The two phrases surrounding with solid lines are sub-phrases which are replaced with nonterminal symbols.

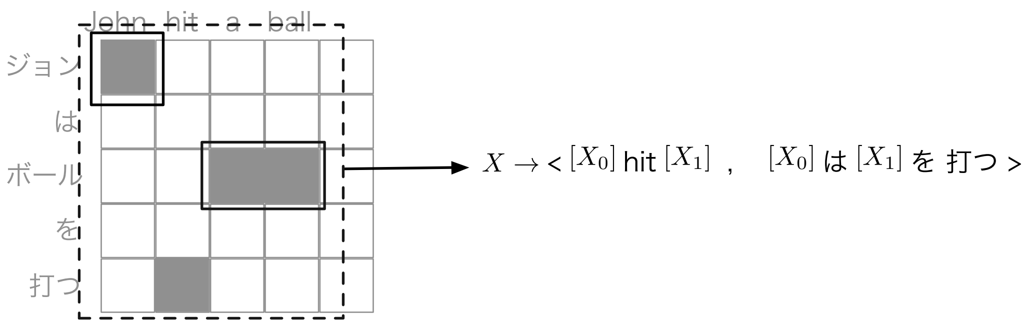

John hit a ball
ジョン
は
ボール
を
打つ
$X \rightarrow < [X_0]$ hit $[X_1]$, $[X_0]$ は $[X_1]$ を 打つ >

Figure 2.8: Illustration of extracting hierarchical translation rules

Following the definition of (Chiang, 2007), a smallest set of translation rules are extracted if they satisfy following requirements:

1) If $< f_i^j, e_{i'}^{j'} >$ is an initial phrase pair, then

$$X \rightarrow < f_i^j, e_{i'}^{j'} >$$

is extracted as a rule.

2) If $X \rightarrow < \gamma, \alpha >$ is a rule and $< f_i^j, e_{i'}^{j'} >$ is an initial phrase pair such that $\gamma = \gamma_1 f_i^j \gamma_2$ and $\alpha = \alpha_1 e_{i'}^{j'} \alpha_2$, then

$$X \rightarrow < \gamma_1 X_k \gamma_2, \alpha_1 X_k \alpha_2 >$$

is extracted as a rule.

Although this scheme only extracts a smallest set of possible translation rules, the number of generated rules usually becomes very large in practice. To tackle this problem, many constrains are applied during the process of rule extraction. For example, a rule must have a least one aligned word pair, the number of non-terminal and the length of initial phrases are limited.

Glue rules

For the robustness of the translation model, a set of glue rules that simply concatenate two sequences is applied. Which has the following form, where S is the start symbol of a sequence.

$$\begin{aligned} S &\rightarrow\ < S_0X_0, S_1X_1 > \\ S &\rightarrow\ < X_0, X_1 > \end{aligned}$$

Model

Hierarchical phrase-based model also applies Log-linear model as its foundation. The features of translation probability become $p(\gamma|\alpha)$ and $p(\alpha|\gamma)$. For each translation rule and glue rule, a penalty feature is also applied.

Decoding

To keep the translation process being finished in acceptable time, the number of nonterminals in each hierarchical translation rule is usually limited to 2. Hence, the decoder is basically running upon CKY (Cocke-Kasami-Younger) algorithm to search for translation hypotheses.

The decoding process begins with phrases containing only one word, for each phrase we enumerate all translation rules, then generate its translation hypothesis and score them. The number of translation in each stack is constrained by Beam search. We do this process recursively until the phrase of whole sentence is reached. For phrases containing two or more words, we check if any sub-phrase containing a portion of them is observed. If so, matched sub-phrases are replaced with gaps X_k to form a hierarchical phrase, then we look for this hierarchical phrase in our translation rules. An example of this decoding process is illustrated in Figure 2.9. From (a) to (d), we observe translation rules for phrases of length 1 to 4 sequentially, then obtain the final translation hypothesis for the whole sentence.

In this case, a possible translation "ジョンはボールを打つ" can be obtained by following derivation, one translation rule is applied in each step.

$$\begin{aligned} < S_1, S_1 > &\Rightarrow\ < X_2, X_2 > \\ &\Rightarrow\ < X_3 \text{ hit } X_4, X_3 \text{ は } X_4 \text{ を打つ} > \\ &\Rightarrow\ < \text{John hit } X_4, \text{ジョンは } X_4 \text{ を打つ} > \\ &\Rightarrow\ < \text{John hit a ball, ジョンはボールを打つ} > \end{aligned}$$

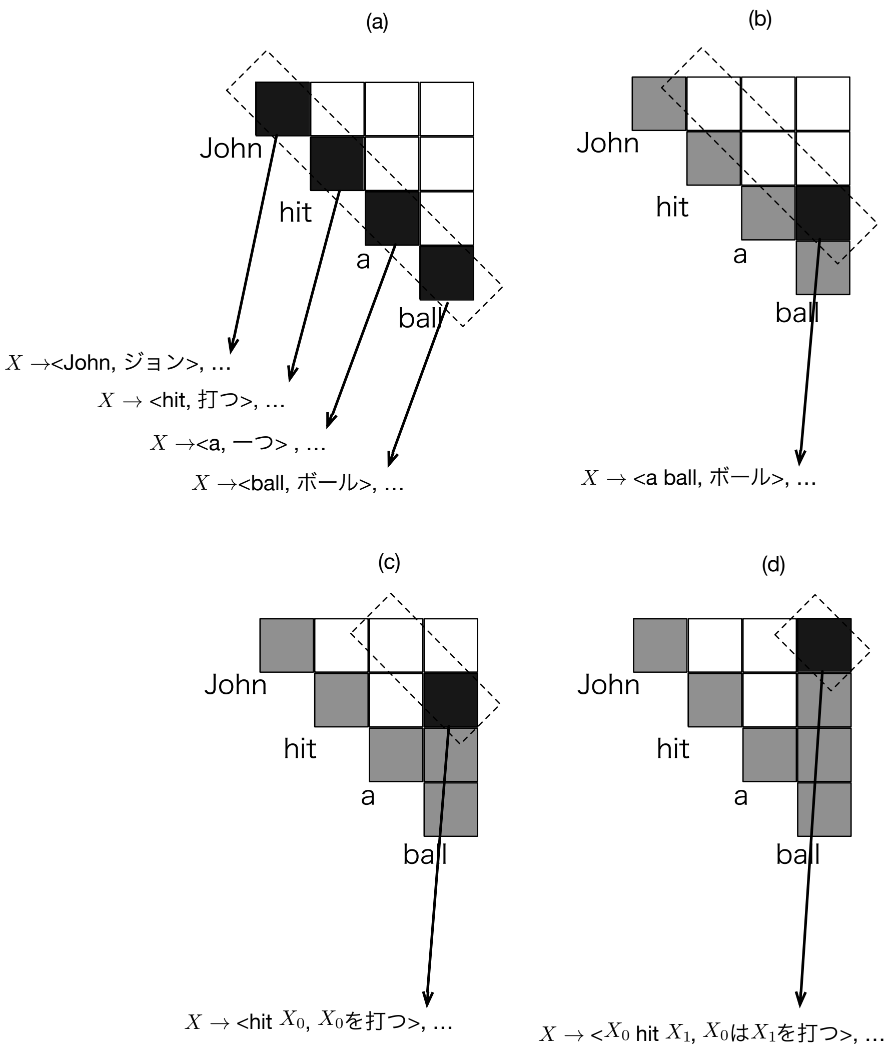


Figure 2.9: Illustration of decoding by CKY algorithm

2.3.5 Evaluation criterions for statistical machine translation

Human evaluation is always the best choice for evaluating machine translation systems. However, in the practical level, human evaluation brings with some problems as it is time-consuming and expensive. Hence, there is need for automatic evaluation.

BLEU (Bilingual Evaluation Understudy) (Papineni et al., 2002) is a evaluation method to satisfy this need, and widely applied in the evaluations for SMT systems. It evaluates machine translated results according to one or more reference human translations, then gives a numerical metric which ranged from 0 to 1. This metric shows the closeness to references. Basically, BLEU computes the concordance rate of unigrams to 4-grams, and applies a so called "brevity penalty" to penalize candidate translations shorter than references.

NIST score (Doddington, 2002) is another automatic evaluation method based on BLEU. In BLEU, all $n$gram correspondences are equally weighted. NIST considers how informative a $n$gram is, which means rarer $n$grams are weighted more than normal $n$grams permeating in the language.

RIBES (Rank-based Intuitive Bilingual Evaluation Score) (Isozaki et al., 2010) is an automatic evaluation method proposed recently which considers correspondence of word order. In RIBES, two word order metrics, Normalized Kendall's τ and Normalized Spearman's ρ are utilized. Evaluations performed on corpus of NTCIR-7 show it has higher correlation with human judgments.

2.3.6 Minimum error rate training

To decode translation models, the parameters λ_m in Log-linear model need to be estimated. However, using Maximum likelihood estimation can not ensure the translation accuracy is maximized. With the emergence of BLEU, a estimation method that directly maximizes this evaluation score is proposed by Och (Och, 2003), which is called Minimum Error Rate Training (MERT).

In this method, an independent development data set $(\mathbf{f}, \mathbf{e})$ is prepared. Given human translated results $\mathbf{r}$ as references and translation results $\mathbf{e}$ generated by machine translation system, the error rate is evaluated by $E(\mathbf{r}, \mathbf{e})$. If we use BLEU as our evaluation criterion, $E(\mathbf{r}, \mathbf{e})$ can be $1-BLEU$. MERT will automatically search for the optimized parameters $\hat{\lambda} = \hat{\lambda}_1^M$ that minimize the sum of error rate between references and best translation hyothesises. That is:

$$\hat{\lambda} = \operatorname*{argmin}_{\lambda} \sum_{i=1}^{|\mathbf{e}|} E(e_i, \operatorname*{argmax}_{h} p_{\lambda}(h|f_i))$$

Here, h in the equation means a possible hypothesis translated from f_i.

Chapter 3

Tree-to-string Machine Translation Models

3.1 Introduction

Same as the Hierarchical phrase-based model, tree-to-string models is another widely researched approach of syntax-based models. Basically, a tree-to-string translation model takes a parse tree (e.g. CFG parse tree or Dependency parse tree) as input, then translate it into string of target language.

Comparing to phrase-based models we described in previous chapter, for tree-to-string models utilize syntactic structures, they generally achieve better reordering especially for long sentences. With benefits from the parse tree, the decoding process is normally faster than phrase-based models, especially when it is compared with hierarchical phrase-based model.

However, the parse tree is also "Achilles heel" of tree-to-string translation models. An accurate and high-speed parser is always required by all phases in SMT from the training phase to the decoding phase. So a trade-off between parsing speed and accuracy is to be considered inevitably. Besides, whether how excellent the translation model is, the translation results of tree-to-string models are always sensitive to parsing errors. Especially those parsing errors that affect large span in a parse tree is critical in decoding. Thus, a approach called Forest-to-string is proposed which utilizes n-best parsing results instead of a golden parse tree, and this gained large improvement in translation quality.

This approach is started by Yamada & Knight (Yamada & Knight, 2001). In their model, CFG parse tree is taken as input, the translation process is carried out for each leaf word in the parse tree (terminal). The tree structure is adjusted to fit target language by two steps, reordering and inserting. In their research, it shows that utilizing syntactic structures helps improving translation quality, especially for language pairs that differ in word order. In (Galley et al., 2006), the translation model is relaxed from the range limit of translation pairs. In their model, not only one single constituent but a treelet (tree fragment) is considered to be a translation unit. In the mean time, Liu et al. also proposed a tree-to-string translation model (Liu et al., 2006), which automatically obtains Tree-to-string

Alignment Templates from training text. Basically, a Tree-to-string Alignment Template is a translation pair that has a fragment of parse tree in source side and plain text in the target side. The evaluation results show that this model outperforms Phrase-based model over BLEU.

As extensions to the approach of tree-to-string, the model of Mi et al. proposed in 2008 utilizes packed forest (Mi et al., 2008), that is roughly a list of ∞-best parse results. Xie et al. proposed a Dependency-to-string translation model in 2011 (Xie et al., 2011), which translates dependency treelets using head-dependents rules. The evaluation results show their model outperforms the Hierarchical phrase-based model over BLEU on Chinese-English translation task, which is the first time a source dependency structure based model catches up with the state-of-the-art models.

In Moses, which is the most influential implementation of state-of-the-art SMT translation models, actually a unified framework is applied for training Phrase-based, Hierarchical phrase-based and tree-to-string models, and decoding them (Hoang et al., 2009). In the training pipelines, despite the phase of syntactic parsing, similar steps are performed for tree-to-string models as other two phrase-based models.

3.2 Syntax rule extraction

The main difference between Syntax-based models and other phrase-based models is in rule extraction. In Syntax-based models, all translation pairs have to correspond to syntactic constituents in the input tree. Normally, even the extraction process is also constrained by tree structures.

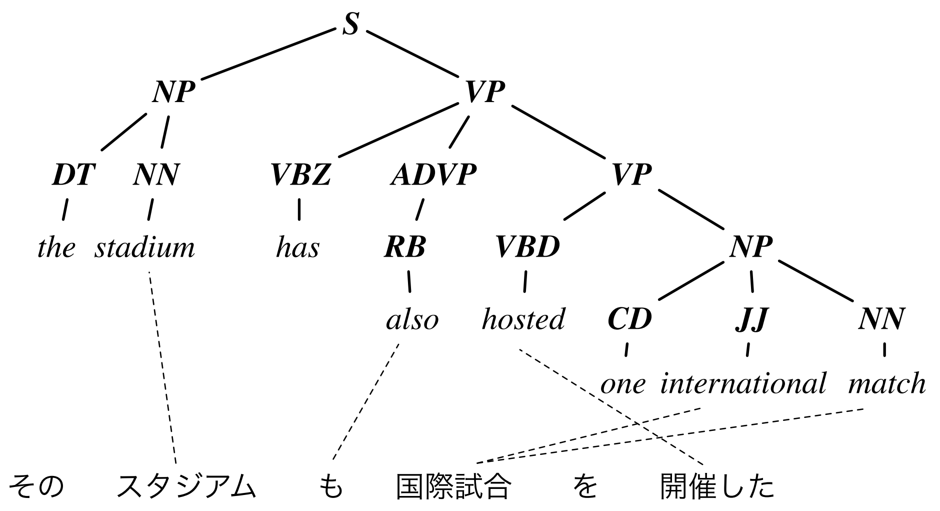


Figure 3.1: An example of parsed CFG tree and corresponding translation

Figure 3.1 shows an example of training pairs which has a parsed CFG tree in source side (English), and plain text in target side (Japanese). Dotted lines in the mid show automatically obtained word alignments. To extract translation rules from this training pair, all tree fragments containing terminals are traversed in the beginning. Basically, as the first step, only those fragments of one height (that is, fragments contains only one layer of child nodes) are extracted. An example of extracted translation pair is shown in Figure 3.2 (a). Most Tree-based models allow extracting composed rules(Galley et al., 2006), in which, several minimal-extracted rules are combined into a larger translation pair. An example of composed rules is shown in Figure 3.2 (b). Next, similar to Hierarchical phrase-based model, if a translation pair exists in another larger translation pair, then it can be replaced by substitution site as Figure 3.2 (c) shows. Tree fragments that only contain non-terminals are also extracted in the form of Figure 3.2 (d) to capture reordering relationships in large span. The positions of substitution sites in target side are decided by the positions of aligned target words.

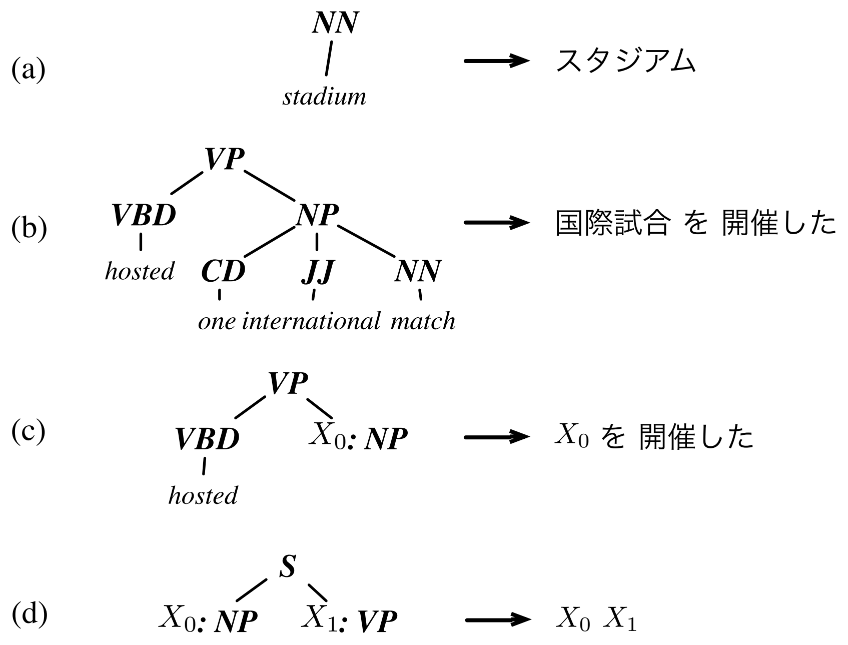


Figure 3.2: Extraction examples of tree-to-string translation model

3.3 Decoding

Typically, chart parsing algorithm is used for decoding Syntax-based models. In the bottom-up fashion, a larger tree fragment is translated recursively, and it produces a partial translation which is also called hypothesis. Also, a new hypothesis is formed by combining small hypotheses with a translation rule. Normally, the translation process of all tree fragments can not be performed simultaneously. In order to fully utilize language model, if we have a rule with "put X_0 into X_1" in source side, all hypotheses in the position of X_0 and X_1 need to be obtained before applying this rule to build a new hypothesis. Although some new methods are proposed to boost decoding speed (e.g. Incremental Decoding (Huang & Mi, 2010)), we only introduce basic concepts of decoding in this thesis.

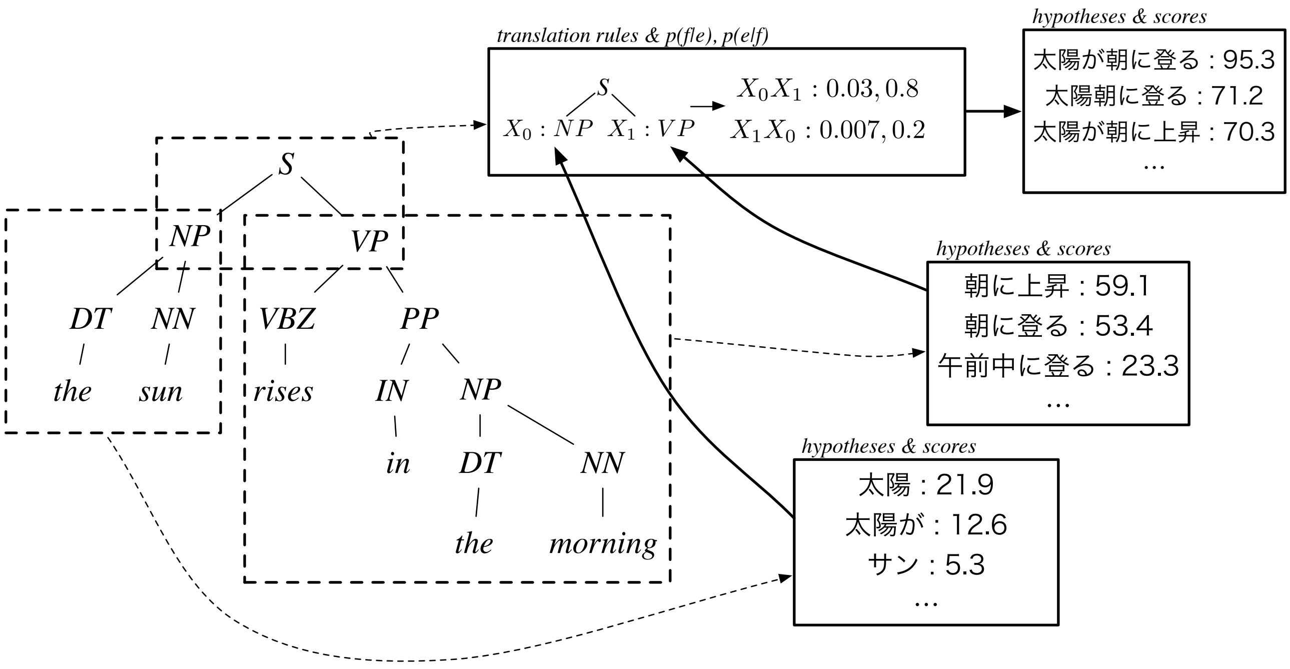


Figure 3.3: An illustration of decoding a tree fragment

An illustration of decoding process in tree-to-string model is shown in Figure 3.3. The input sentence is "the sun rises in the morning", and a parsed CFG tree is shown in the left side. We first decode all leaf nodes in the tree, then hypotheses that covering a larger span are built in each step. In Figure 3.3, now we have obtained the hypotheses of the span "the sun" and "rises in the morning" respectively. By Log-linear model, each hypothesis is assigned with a score. The next and also the final step is to translate the fragment in the top of parse tree. We first enumerate all translation rules we obtained in the extraction phase. As the two child nodes are both non-terminals, only two rules are to be considered ("X_0X_1" and "X_1X_0"). So basically, these two rules are functioning to concatenate two partial translations either in a forward or reversed order. By applying these rules, we form new hypotheses for the whole sentence, and the score for each hypothesis is recalculated by Log-linear model. Finally, the hypothesis with the highest score is selected to be the golden translation result.

3.3.1 Cube Pruning

Considering the step of building new hypotheses in Figure 3.3. Assuming we have n_1 and n_2 hypotheses in two hypothesis stack corresponding to $X_0 : NP$ and $X_1 : VP$ in the tree fragment, then we obtained n_r translation rules to combine them. If we expect to evaluate all possible combination of those rules and hypotheses, then we need to perform $n_r \times n_1 \times n_2$ times of calculation. Even if we only have 100 hypotheses in each stack and 100 rules obtained, the number of possible combinations will be 1,000,000. As we illustrated in Figure 3.3, good translations tend to be form by translation rules and hypotheses having good score. Obviously, it is redundant to search for all possible combinations. Cube Pruning (Chiang, 2007) is a method to tackle this problem.

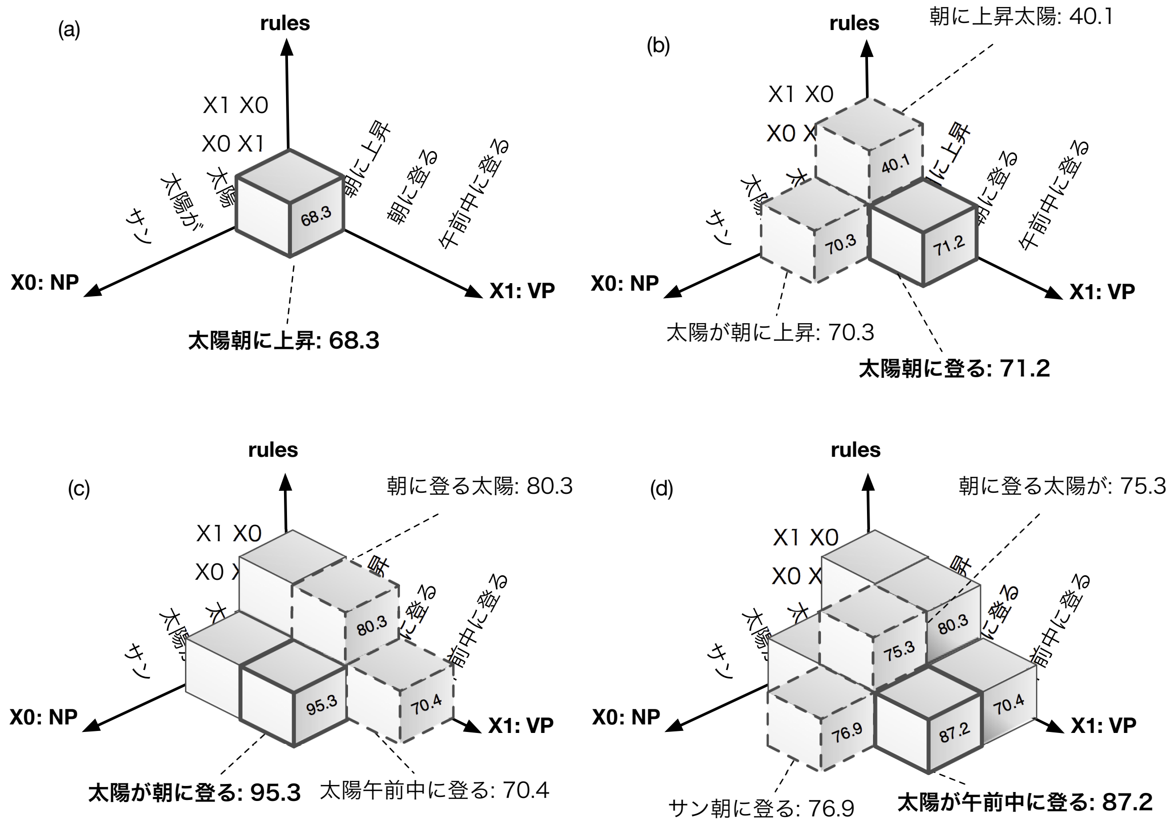


Figure 3.4: An illustration of cube pruning

In the above case, we first build a three-dimensional coordinates, so as to mark the position of each hypothesis. We sort rules and child hypothesis by score, and put them to each axis respectively. Rules and hypotheses with higher scores have lower coordinate values. As the initial step, we combine the first rule and hypothesis in each axis to form an initial hypothesis, and place it to the position (0,0,0). This is shown in Figure 3.4 (a). Next, we expand our searching area with one block in each direction, obtain the hypotheses at (1,0,0), (0,1,0) and (0,0,1). In these three hypotheses we examined in this step, the hypothesis "太陽朝に登る" has highest score. This step is shown in (b). Then next expansion starts from the best hypothesis we searched in last step, we do this recursively as (c) and (d) show. However, the number of hypotheses to be observed is limit by a parameter called Cube size.

Cube pruning is crucially important for decoding Tree-based translation models, because in non-binary parse tree, a node may have multiple non-terminal and terminal children. If we just limit the size of hypothesis stack and rules, that means only a small portion of entire set of combinations is observed. In Cube pruning, the searching area which is more likely to produce good translations will be expanded efficiently.

Chapter 4

Improving Translation Quality by Tree Combination

4.1 Parse tree deficiencies

Context-free grammar (CFG) parse trees are widely used in tree-to-string statistical machine translation models and generally provide comprehensive linguistic information, including the structure of a given natural language sentence. On the other hand, Dependency parse trees provide semantic information about the relations between words, which could be helpful for capturing long-range word relationships in long sentence translation tasks.

Although Dependency trees provide more information about relationships between words that are far apart, most Dependency tree parsers struggle to correctly parse non-content words such as function words, idiomatic usages and some symbols that have no explicit semantic definitions. These types of words could be considered as the constituent parts of sentence structure. Unless these words are correctly translated, the translation of the overall sentence will have problems.

We give an example in Figure 4.1 to demonstrate this problem. The Dependency tree shown in Figure 4.1 corresponds to the sentence "do not undertake a project unless you can implement it," which translates into Japanese as "あなたがそれを実施することができない限り、プロジェクトを引き受けないでください." Usually, " X_0 unless X_1" produces the translation "X_1 できない限り、 X_0" in Japanese and is independent of the words in X_0 and X_1. Hence, the word "unless" can be considered a word that determines sentence structure. As we discussed in Chapter 3, for tree-based models, normally a parsed tree is decoded in bottom-up fashion. Hence, the word "unless" will be translated at the beginning, as it is a leaf node. This decoding makes the clues of the overall sentence structure hard to capture and translate accurately.

This problem could be mitigated if the input of the translation system were to combine both CFG and Dependency parse trees. In fact, the method described in this thesis will yield a tree in the style of a hierarchical phrase tree, in which each node represents a phrase with several non-terminal symbols. Each node in the combined trees has at least one terminal. For each node, we pick out a terminal to be the headword of the node, and then use its

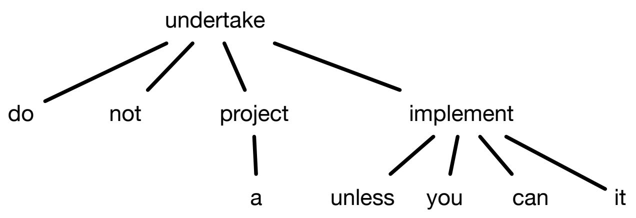


Figure 4.1: Parsed Dependency trees may cause translation problems

linguistic tag to represent the type of node.

4.2 Combining dependency relations with phrase-level structures in CFG trees

The core concept of the method described in this thesis is to construct a new tree that intuitively keeps dependency relations at a large scale as well as reasonable phrase-level structures extracted from the CFG tree. This method could also be thought of as reconstruction of a CFG tree using dependency relations because the new tree is also a CFG tree. Nodes close to the top in the Dependency tree also tend to be close to the top in combined trees. This can make it easier for the translation model to capture translation relationships between languages that differ in word order, as the head-dependent relationships of words are usually in accordance, even if the order differs.

We apply heuristics to perform this reconstruction, which will be described in following sections. In Sections 4.2.1 and 4.2.2, we first simplify the tree structure to handle words without dependency relations. Then, Section 4.2.3 describes the core part of our combining method, which incorporates the dependency relations into the CFG parse tree by repositioning terminal nodes. Finally, the method for determining non-terminal tags is described in Section 4.2.4.

4.2.1 Removing redundant non-terminals in CFG trees

We simplify the parsed CFG tree by removing redundant non-terminals because keeping too many non-terminals in a parse tree does not improve the translation quality. Our method accomplishes this in two steps.

First, we remove non-terminals that have only one child node, as shown in Figure 4.2. In this step, we avoid translating ambiguous non-terminals that represent the linguistic tag of a terminal or a subtree.

The second step is to eliminate layers that contain no terminals (here, a layer is formed of nodes that belong to the same parent node). The existence of such layers sometimes misleads the decoding process because the translation of a tree fragment consisting of pure non-terminals is ambiguous. We expect at least one terminal in an arbitrary tree fragment

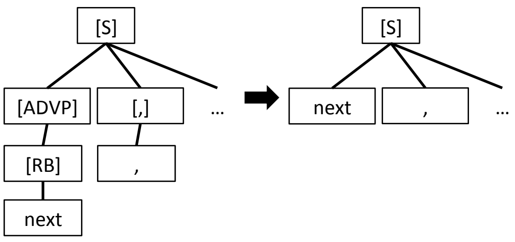


Figure 4.2: Removing non-terminals that have only one child node

to provide a clue for translation.

4.2.2 Separating words without dependencies

Some words do not appear in the parsed Dependency trees, so we place these words into separate layers. These words are usually symbols that represent the structure of the sentence, so they should be separated from the content words that represent meanings. Although it depends on what parser we use, it would normally introduce an ambiguity if we were to force a dependency head on these words.

This step is done in the preprocessing phase. If a layer contains terminals without dependency relationships, we do not allow two continuous non-terminals appear in the same layer. For these continuous non-terminals, we create a new non-terminal with the tag “X” and move them so that they are the children of the newly created non-terminal. The tag “X” will eventually be replaced, as we apply dependency head tags to the non-terminals as described in section .

The most common example of this is a comma in English. In figure 4.3, we show a example of separating a comma, where “[NP] [VP]” are continuous non-terminals that appear in the same layer as the comma. We create a new non-terminal “X,” and then move “[NP] [VP]” beneath it.

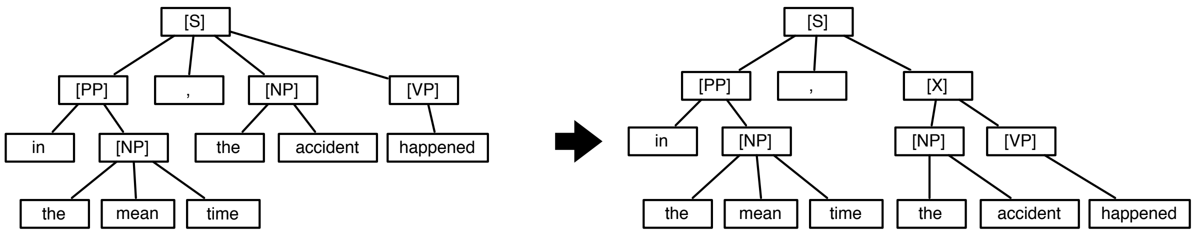


Figure 4.3: Comma separation example

4.2.3 Repositioning nodes by dependency relationships

In the above case, when a layer contains only one non-terminal, we simply remove this node and directly concatenate its child nodes to its parent node. In a more complex case, a layer is composed of two or more non-terminals. In this situation, a non-terminal node is selected for replacement by its child nodes. We perform this selection by counting the internal dependencies in this layer and finding the non-terminal with the subtree that has the most dependent nodes in the scope. This means that the subtree should contain a semantic head for the entire layer. This step is illustrated in Figure 4.4.

Figure 4.4 shows a fragment with a internal layer formed from non-terminals. We count the dependent words for all child nodes and mark the word with the most dependent words as the semantic head. Then we replace the non-terminal above the semantic head with its child nodes.

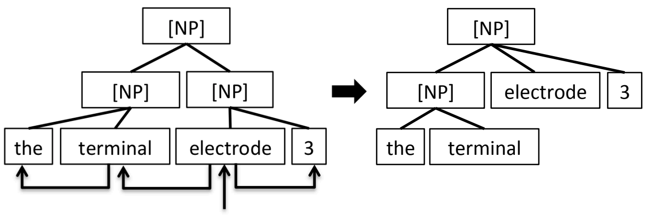


Figure 4.4: Eliminating a layer with multiple non-terminals

In Figure 4.5, we show another example of restructuring a tree fragment that contains a noun and verb. As the verb "depends" is the head of this span, we lift it up to the layer immediately beneath the top node. For this kind of verb, we also put its prepositional modifier in the same layer. In this example, the prepositional word "on" is moved to the same layer as "depends."

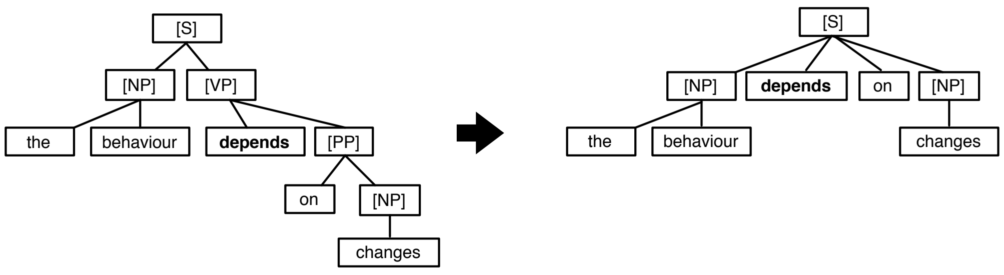


Figure 4.5: Restructuring a tree fragment that contains a verb as a head

When the above steps are finished, we can merge the nodes in each layer together into one node to form a tree in the style of a hierarchical phrase tree, which gives a more intuitive image of the translation process.

The repositioning method discussed in this section generally attempts to put dependency heads into tree fragments that cover larger spans. Hence, the translation model can more easily capture long-range translation rules.

4.2.4 Tag determination by headwords

As the structure of the parse tree is adjusted, the tags of non-terminal symbols need to be reconsidered. In our method, new tags are determined by the corresponding headwords of child nodes.

The headword of a phrase can be determined using dependency relations. In contrast to some other methods (Li et al., 2012), we allow a phrase to have only one headword. Given a token sequence f_n^m in the sentence f_i^j that is formed of terminals and non-terminals and covers a span of words f_i^j, we define the headword in the following way.

Definition 1. *For token sequence f_n^m, a terminal $f_k \in f_n^m$ is regarded as the headword if f_k has the most dependent words in f_i^j.*

If two terminals have equal numbers of dependent words, we select the right-most word as the headword. A combined tree fragment with marked headwords and new tags is illustrated in Figure 4.6. In order to facilitate the understanding of rule acquisition in the next section, we put the nodes of the same layer into one box here. The headword for each node is marked with an underscore.

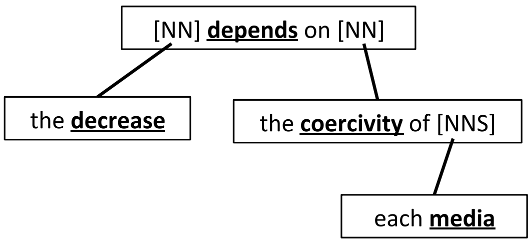


Figure 4.6: Example of a tree fragment with marked headwords and converted tags

4.3 Rule Acquisition

We extract the minimal necessary translation rules from pairs formed from a combined source-side tree and target-side string.

Given a combined tree and words with alignment relations A, for an arbitrary subtree t and target-side words $e_i^j = e_i, e_{i+1}, ..., e_j$, we extract this pair as an initial translation rule if the following conditions are satisfied.

1) $\exists\ f_m \in t, e_n \in e_i^j \ \ s.t. \ \ < f_m, e_n > \in \ A$

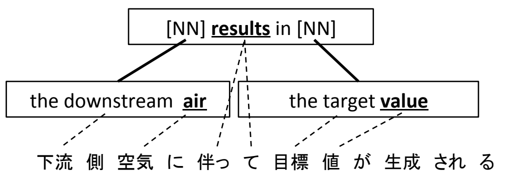
2) $\forall\ f_m \notin t, e_n \in e_i^j \ \ , \ \ < f_m, e_n > \notin \ A$

3) $\forall\ f_m \in t, e_n \notin e_i^j \quad , \quad < f_m, e_n > \notin\ A$

4) e_i^j has minimal length

If a tree fragment in the initial translation rules contains other fragments that also exist in the initial translation rules, then we mark those child fragments as sites constrained by the linguistic tag of the headwords of the child fragments. Corresponding sites on the target side are not constrained.

An illustration of hierarchical rule extraction is shown in Figure 4.7. We show a combined tree and its corresponding Japanese translation. The dotted lines show word alignments automatically retrieved. According to the alignments, we know the two child fragments have corresponding translations “下流 側 空気” and “目標 値” respectively. Hence, we can replace either fragment with a site X_1 to form two hierarchical rules. In addition, if we replace both fragments with X_1 and X_2, the third hierarchical rule shown in Figure 4.7 can be formed.



[NN] results in [NN] ➔ X_1 に 伴っ て X_2
the downstream air results in [NN] ➔ 下流 側 空気 に 伴っ て X_1
[NN] results in the target value ➔ X_1 に 伴っ て 目標 値

Figure 4.7: Hierarchical rule extraction

4.4 The model

As in other recent statistical machine translation models, we adopt a general log-linear model:

$$\hat{e} = argmax_e \prod_i \phi_i(e)^{\lambda_i} \quad .$$

Here, we use following features for decoding:

- Translation probabilities, $P(e|f)$ and $P(f|e)$
- Lexical translation probabilities, $P_{\text{lex}}(e|f)$ and $P_{\text{lex}}(f|e)$
- Language model, $P_{\text{lm}}(e)$

- Rule penalty, $\exp(-1)$
- Word penalty[1], $\exp(-1)$

These features are basically the same as those used in the Hierarchical phrase-based model.

4.5 Decoding

In our decoder, we use bottom-up decoding to find the best derivations. Given a combined tree, for each node n we attempt to find translation rules that exactly match the one-layer tree fragment rooted at n. If any rule is found, we traverse all tree fragments rooted at n for a maximum of three layers to find translation rules that cover a larger span. If no rules are found in these steps, we attempt to find composed rules. As a last resort, we decompose all child nodes in the fragment and apply a glue rule to concatenate their translations together.

The search for translation rules for the top fragment in Figure 4.5 is illustrated in Figure 4.8. In our implementation, we do not restrain the searching process by the tag of a parent node, so the root tag is presented by "X." For the given tree fragment, we search for rules that exactly match the fragment in the source side, which is "[NP] depends on [PP]." If any matched rules exist, then it is likely that rules that match a larger portion of the entire tree exist. Hence, we try to expand the two non-terminals "NP" and "PP" to gain larger fragments and find corresponding rules. If we fail to find exact rule matches, we decompose the fragment into several portions and find translation rules for them accordingly. Then, we use a glue rule to concatenate the translation hypotheses together. If no rules found in the above steps, we search for "[NP]," "depends," "on," and "[NP]" individually. As we decode in a bottom-up fashion, the hypotheses for translating non-terminals have already been prepared. For terminals, we allow an English word to be translated to itself if no Japanese translations are found in the rule table.

We use cube pruning (Chiang, 2007) to find the best derivation for each tree fragment, and also set a beam size for every hypothesized stack of nodes.

4.5.1 Optimizing for memory usage

Normally, the decoder of a statistical machine translator reads the whole rule table into memory at the beginning. This requires a large amount of memory even though most of the translation rules are not used during the translation process. In order to address this problem, we hashed all source-side strings of the translation rules, and loaded only those hashed strings into memory. As a result, our decoder consumes only 3.8GB of memory during the evaluation task.

Our decoding process is divided into two steps: rule fetching and translation. In the rule fetching step, we find all possibly usable translation rules for each node in the combined tree, and dynamically read them into memory from the on-disk rule table. In the translation

[1]We did not include the word penalty feature in our translation model for the formal run of the NTCIR-10 workshop.

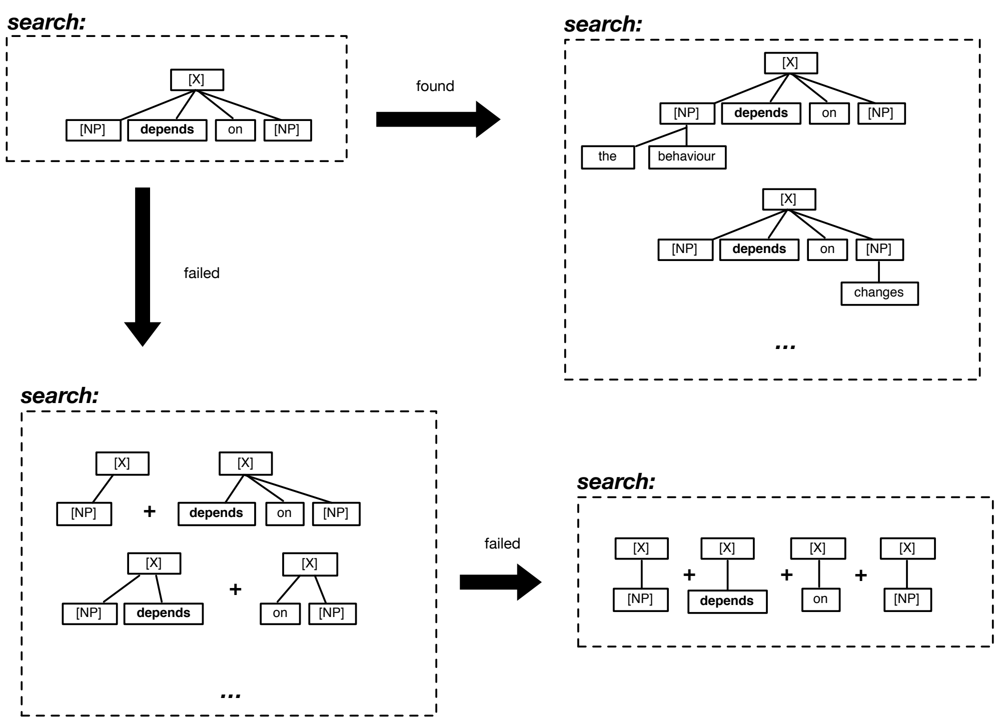


Figure 4.8: Search for the translation rules of a tree fragment

step, as all translation rules have been prepared, there are no I/O operations. So we could perform these two steps in two parallel processes, which will increase the speed of decoding.

Chapter 5

Experiments

In this chapter we present several evaluations carried out with our proposed model. In section 5.1, we present the evaluations in NTCIR-10 workshop. In their official human evaluations, the translation results produced by our model are shown to be more adequate comparing with their baseline models (Hierarchical phrase-based model and Phrase-based model). In the evaluations carried by ourselves (section 5.2), it shows our model generally produces better translations comparing with translating CFG parse trees alone. In the rest parts of this chapter, we present the error analyses of some samples picked out from translation results and an attempt to utilize hybrid parse tree in the translation model. This chapter ends with a sample of combined parse tree and several translation samples.

5.1 Evaluations at NTCIR-10 workshop

The NTCIR Workshop is a series of evaluation workshops designed to enhance research in information access technologies including machine translation, which is held by NII (National Institute of Informatics). In NTCIR-10, the organizer of English-to-Japanese Patent Machine Translation Task provides a training corpus which contains 3M bilingual sentences, 2000 bilingual sentences for tuning the model. In the formal run of the evaluation, participants are required to translate 2300 sentences and submit the results to the organizer. They evaluate the results using both automatic metrics and also human evaluation methods proposed by Goto et al..

In the preprocessing phase, we used Mecab (a part-of-speech and morphological analyzer for Japanese) to segment sentences into words in the Japanese side of the corpus.We parsed English-side sentences with the Berkeley Parser (Petrov et al., 2006) to get CFG parse trees, and then produced normal dependency parse trees with the Stanford Parser (Klein & Manning, 2003) using parsed CFG trees from the Berkeley Parser as input.

We generated word alignments using GIZA++ on the training corpus in both directions with the "grow-diag-final-and" refinement.

For the language model, we obtained a 5-gram model on Japanese-side sentences with interpolated Kneser-Ney smoothing using the SRILM Toolkit (Stolcke, 2002).

In order to tune the weights of each feature in our translation model, we used Minimal

Error Rate Training. The development data contains 2000 bilingual sentences.

In the decoding phase, we used 2000 for the cube size for cube pruning, and 1000 as the beam size.

To solve the problem of the sparseness of the rule table, we simplified the linguistic tag set produced by the Berkeley Parser from 45 types to 8 types. We generally mapped noun tags like "NNS" and "NN" to a single tag "NP". Verb tags were mapped to "VP". Some rare linguistic tags like "SYM" were also mapped to "NP". Finally, we adopted 8 tags in our translation system:

NP, VP, JJ, CD, IN, S, TO, DT .

In the intrinsic subtask of the NTCIR-10 PatentMT English-to-Japanese task, we submitted three run results: TSUKU-ej-int-1, TSUKU-ej-int-2 and TSUKU-ej-int-3 (Zhu et al., 2013).

TSUKU-ej-int-1 was translated using a normal 5-gram language model, which was trained from the same bilingual corpus we used to train our translation model. While the other two results are translated using a 5-gram language model trained from a large-scale monolingual resource of about 174M sentences from Japanese patent applications in the period 1993–2000.

We used KenLM in the decoding process for TSUKU-ej-int-1 and TSUKU-ej-int-3, and the LSH compressed language model proposed by Norimatsu for TSUKU-ej-int-2.

5.1.1 Formal run results

The official automatic evaluation results are shown in Table 5.1. Our three runs were worse in terms of BLEU but better in terms of the NIST and RIBES metrics than the organizer baseline using a hierarchical phrase-based model.

Table 5.2 shows the subjective evaluation results. Our translation of TSUKU-ej-int-1 achieved a better Adequacy score than the organizer baselines.

Table 5.1: Official automatic evaluation results for the English-to-Japanese task

System	BLEU	NIST	RIBES
System runs			
TSUKU-ej-int-1	0.3141	8.126	0.7555
TSUKU-ej-int-2	0.319	8.1894	0.7565
TSUKU-ej-int-3	0.3176	8.1769	0.7566
Organizer baselines			
BASELINE HPBMT	0.3298	8.0837	0.7231
BASELINE PBMT	0.3361	8.1816	0.7042

Table 5.2: Official subjective evaluation results for the English-to-Japanese task

System	Adequacy	Acceptability
System runs		
TSUKU-ej-int-1	2.7933	0.4088
Organizer baselines		
BASELINE HPBMT	2.69	-
BASELINE PBMT	2.5333	-

5.1.2 Post-evaluation results

The results of the formal run were not ideal because we did not include a word penalty in the features of our model due to an oversight. Table 5.3 shows the post-evaluation results. The scores are slightly different to the official automatic evaluation scores because we are using "mteval-v11b.pl" to do the automatic evaluation. TSUKU-ej-int-1^{submitted} is the translation result we submitted in the formal run and TSUKU-ej-int-1^{post} refers to the translation result of our post-evaluation.

By appending word penalty to the features of our model, we could see a promotion in BLEU score but not too varied according to other evaluation criterions.

Table 5.3: Post-evaluation results for the English-to-Japanese task

System	BLEU	NIST	RIBES
System runs			
TSUKU-ej-int-1^{post}	0.3308	8.014	0.756
TSUKU-ej-int-1^{submitted}	0.3145	8.0821	0.756
baselines			
BASELINE HPBMT	0.3306	8.0849	0.7242

As we are expecting our model to achieve better translation quality in long sentence translations, we also evaluated it with a test corpus formed by 1048 sentences, which are picked out from formal run data of NTCIR8 and NTCIR-10 and each sentence is longer than 40 words in English side.

Evaluation results in table 5.4 show our model could generally generate better translations than that of hierarchical phrase-based model. Conversely, short sentence translation is the weakness of our model due to the constrain of parsing errors.

Table 5.4: Post-evaluation results for the long sentence translations

System	BLEU	NIST	RIBES
System runs			
TSUKU-ej-int-1^{post}	0.3238	7.966	0.7141
Baselines			
BASELINE HPBMT	0.3185	7.9946	0.6718

5.2 Evaluations comparing with the CFG-based SMT systems

To compare with tree-to-string translation models using unmodified CFG trees, we built two baseline models. The first one is Syntax-based translation model implemented in Moses. In our experiments, we tried both right-binarized CFG tree and non-binary CFG tree as the input. We also built another model which translates non-binary CFG tree using our self-made decoder. In Table 5.5, we show the evaluation results of above models.

Table 5.5: Self evaluation results

Model	BLEU	RIBES
Moses Tree-to-string using non-binary CFG	0.2855	0.684
Moses Tree-to-string using binary CFG	0.3273	0.724
Self-made decoder using non-binary CFG	0.2865	0.728
Self-made decoder using combined tree (Proposed model)	0.3308*	0.756

* means the result is significantly different at 5% level comparing with other methods (only for BLEU evaluation criterion).

5.3 Error analysis

We analyzed 16 translation samples produced by our model that contain errors in them. 5 of them are caused by parsing error, 8 of them is due to the error produce by translation model, and 3 samples are caused by inappropriate preprocessing for escaped HTML characters.

Parsing errors are inevitable in syntactic parsing, and caused by multiple reasons. A typical example of parsing error in the domain of patent is shown in Figure 5.1. This CFG tree is parsed from the sentence "The fixing roller 51 rotates clockwise as indicated by the arrow.". In this Figure, Non-terminal tags are presented in parentheses, all leaf nodes contains terminals. Both Berkeley parser and Stanford parser produce similar result. This error is caused by the existence of terminal node "[CD]51", which should be a portion of "fixing roller 51". This kind of parsing error prevails in parsing results of patent documents.

For errors involved in translation model, an example is shown in Figure 5.2. This tree is parsed from the sentence "this makes it much easier to secure the soldering area", and has the Japanese translation "これにより、はんだ付け面積を更に確保し易くなる". In the left side, we show a tree combined CFG and Dependency relationships by our proposed method, with the finally applied translation rule of each minimum fragment for the best hypothesis shown in the right side. In this example, no parsing error is involved, but the best translation hypothesis gives the result "この半田付け領域確保も容易に行うことができる". We consider this is caused by failing to capture the translation relationship of "this makes it $X_0 \rightarrow$ これにより、X_0", which is a easy job in phrase-based models. The tree fragment covers the phrase "this makes it" contains 3 layers, so it's generally a intractable problem for Tree-based models.

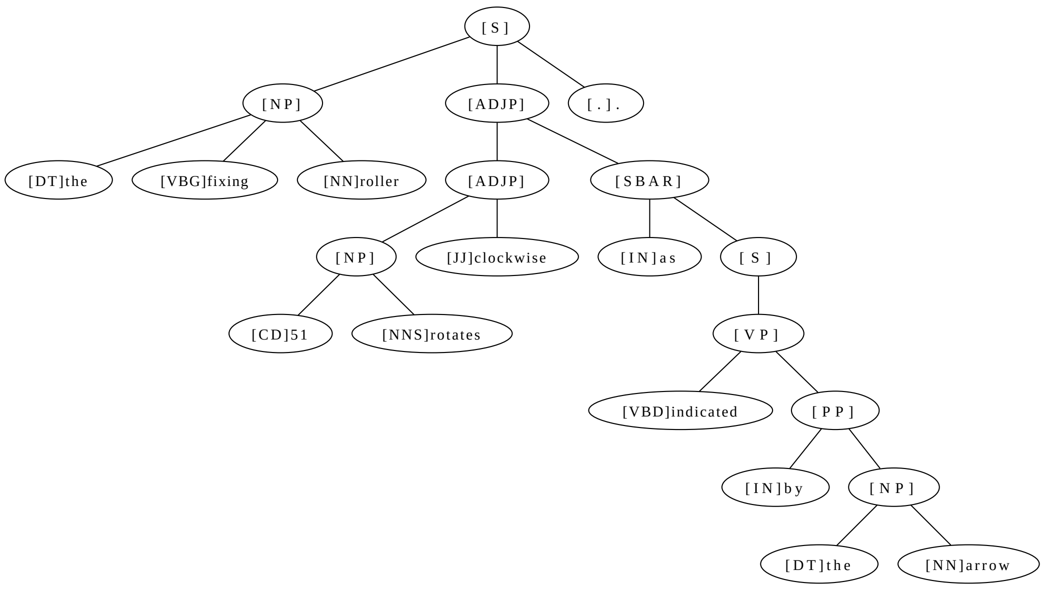


Figure 5.1: A typical example of parsing error in patent

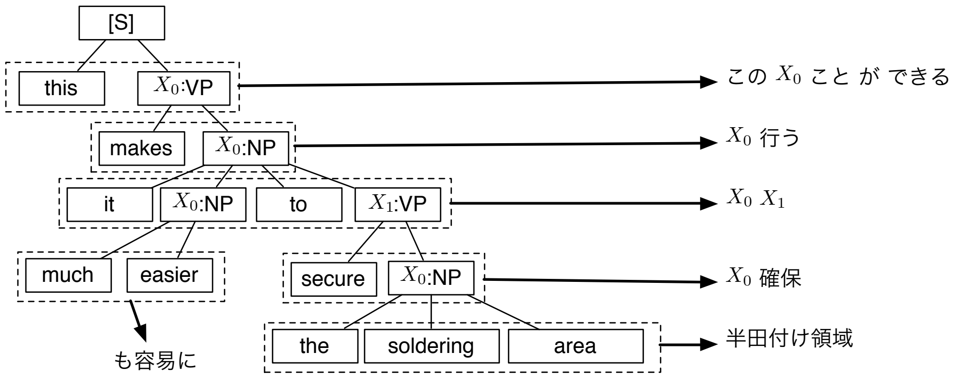


Figure 5.2: A combined tree and best hypotheses for each node

5.4 Translating hybrid parse trees

As we discussed in Chapter 3, our model is also very sensitive to parsing errors. Although Forest-based decoding is a solution to tackle this problem, it will significantly increase decoding time. We tried a different approach to perform a hybrid translation which translates parse trees of multiple different parsers. In our experiment, Berkeley parser and Stanford parser are selected. According to the work of (Cer et al., 2010), a comparison of Unlabeled and labeled attachment F1 score to generate standard Stanford Dependencies for Berkeley parser and Stanford parser is shown in Table 5.6.

Table 5.6: Unlabeled and labeled attachment F1 score (%) of Berkeley parser and Stanford parser

Parser	Unlabeled	Labeled
Stanford	87.2	84.2
Berkeley	90.5	87.9

We train our model upon the parsing results of Berkeley parsers, but in the decoding phase, we parse one sentence using two parsers respectively. Thus we have two set of parse trees, we decode them simultaneously, and then select the translation hypothesis with the highest score for each input sentence. The evaluation result compared with Hierarchical phrase-based model and proposed model that decodes parse trees of Berkeley parser is shown in Table 5.7.

Table 5.7: Evaluation results of proposed model with hybrid parse trees

Model	BLEU	RIBES
Moses Hierarchical phrase-based model	0.3306	0.7242
Proposed model decodes parse trees of Berkeley parser	0.3308	0.756
Proposed model decodes hybrid parse trees	0.3374	0.761

5.5 Sample of Tree combination

In this section we show a sample of tree combination. In Figure 5.3, we present a sample of CFG parse tree, where the number in each node helps to distinguish between nodes with same tag. The combine tree correspond to the same sentence is shown in Figure 5.4. Where the words containing the head are putted into a higher layer comparing with them CFG tree.

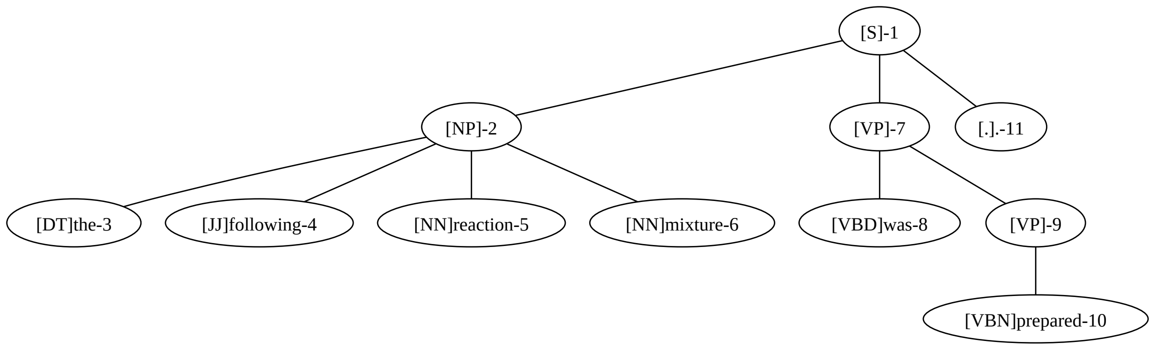


Figure 5.3: A sample of CFG parse trees

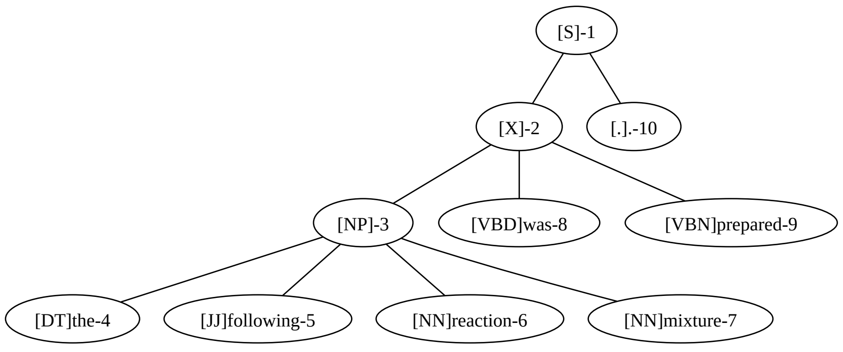


Figure 5.4: A sample of combined parse trees

5.6 Samples of translation

In table 5.8 we show a improved translation sample comparing with Hierarchical phrase-based model. We can observe from the translation results that the translation result of Hierarchical phrase-based model is affected by the order of words in English side sentence. Our proposed model captured the head "calculated / 算出した" (words with underlines) and put it to the end of Japanese translation correctly.

Table 5.8: A translation sample comparing with Hierarchical phrase-based model

English input	Specifically , the FF amount is calculated at every second calculation timing based on an average value of fuel injection amounts , as will be described later .
Result of HPB model	具体的には、ＦＦ量を第２の演算タイミング毎に算出された燃料噴射量の平均値に基づいて、後述するようにされている。
Result of proposed model	すなわち、ＦＦ量は、後述するように燃料噴射量の平均値に基づいて第２の演算タイミング毎で算出した。

Another sample of the translation result of our proposed model comparing with translating CFG trees (using Moses as decoder) alone is shown in table 5.9. The tree-to-string translation model translating CFG trees alone is still unable to capture and translate the whole sentence correctly. With the correct recognition of the head "arranged / 配設されている", our model produced a significantly better result.

Table 5.9: A translation sample comparing with translating CFG trees alone

English input	In this embodiment , the airbag apparatus 120 is arranged so that the direction of protrusion (deployment and inflation) of an airbag 122 (described later) which constitutes the airbag apparatus 120 is directed toward a lumbar part area 132 of the rider protecting area 130 .
Result of translating CFG trees	本実施形態では、エアバッグ装置１２０が配置されている ようにある方向の突起（膨出展開）の、エアバッグ１２２は、エアバッグ装置１２０を構成する（後述する）を向けて腰領域１３２の、ライダー保護領域１３０。
Result of proposed model	本実施例では、このエアバッグ装置１２０がエアバッグ１２２（後述）はこのエアバッグ装置１２０（膨出展開）の突出方向を腰椎部運転者保護領域１３０の領域１３２に向かうように配設されている。

Chapter 6

Conclusions

In this thesis, we have described the SMT translation model that uses as input a combination of CFG and dependency parse trees. The combined trees maintain dependency relations by adjusting node positions according to their dependency heads, and defining headwords in each combined phrase-like node to provide linguistic tags. As our combined trees are transformed basing on parsed CFG trees, the structure words remain in reasonable positions.

Our experiments focus on the English-to-Japanese translation. As English and Japanese hugely differ in syntactic structure and word order, translation models that do not utilize head-dependent relationships tend to be affected by the syntactic structures or word order of English. Especially in the translation tasks for long sentences, they can hardly capture the head of the sentence (usually a verb), and put it to the end of Japanese translation. In our experiments, it shows combining dependency relationships into CFG parse trees benefits the translation model to enable it to capture head words and translate it correctly, even in a long sentence.

We implemented a Tree-to-string decoder to carry out several experiments to evaluate our model, which show our translation model using combined parse trees generally produces better translation results comparing with translating CFG parse trees alone. In the human evaluation carried out in NTCIR-10 Workshop, the results show our model produces more adequate translations comparing with the state-of-the-art Hierarchical phrase-based model and Phrase-based model.

By the error analysis of some sample translations of our model, it revealed the parsing errors are still affecting the translation accuracy in our model. Also, for some idiomatic phrases, it is difficult for our model to capture and translate them correctly.

Acknowledgments

The finish of this thesis is impossible without the constant support of Prof. Yamamoto. Sometimes I recall the beginning of my master course, I had strong motivation and enthusiasm to do some researches in the field of NLP. Before enrolling in University of Tsukuba, I considered a bunch of research topics, which include chatbot, question answering machine and a system which can automatically retrieve human knowledges from Web. So basically I wanted to make a intelligent system more than so-called research, let alone researching in the field of Machine Translation. In the beginning period, it was tough for me to realize the research is totally different from what I thought. At that time, Prof. Yamamoto gave me many concerns and guidances, which led to the decision of researching Machine Translation. I believed it could be an interesting field for me and so it did. The materials of his lectures also benefited me a lot as I started from zero in this field. When I got some inspirations or ideas in my research, even they are immature, he was also glad to discuss them with me. Those pieces became the accumulation and production of knowledge for me. Here, I would like to express my sincere gratitude to Prof. Yamamoto, I could not imagine having a better advisor.

I also want to thanks Jun-ya Norimatsu, who is currently a PhD student in the NoW (NLP on the Web) lab of University of Tsukuba. He always makes all-out effort to help me both in research and also in faith. He is a man of principle, I regard him as a safe haven that I can always rely on. While I was implementing a decoder (translation system), I faced many challenges as that was a hard work and any miss could affect the performance of the decoder. At that time, even he was research in the field of language model, he tried best to recall what he did several years ago and help me to solve some critical problems in the implementation. Also, this thesis benefits a lot from his advices. Sincerely thanks.

Special thanks go to my best friends, Yasuo Kato and Oscar Garcia. We kept a prayer meeting in every Thursday for about half a year till now. Many times, I was encouraged and refilled in the prayer with them. Many times, I was enlightened from the low valley of my life because of their enthusiastic prayer. As there is a song from my favorite band Secret Garden, which is named "Sometimes a prayer will do". Without their support, I don't know where can I live and what way should I go. As Ephesians 5:8 tells, "For you were once darkness, but now you are light in the Lord. Live as children of light[1]", my concerns are allayed when I return to my spiritual home.

Thanks organizers of PatentMT task of NTCIR(NII Testbeds and Community for Information access Research) Workshop for providing bilingual patent data, my research is benefited a lot from it.

[1]Citing from New International Version Bible

Bibliography

[1] Noam Chomsky. Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124, 1956.

[2] Hieu Hoang, Philipp Koehn, and Adam Lopez. A unified framework for phrase-based, hierarchical, and syntax-based statistical machine translation. In *Proc. of the International Workshop on Spoken Language Translation, Tokyo, Japan*, pages 152–159, 2009.

[3] Jun Xie, Haitao Mi, and Qun Liu. A novel dependency-to-string model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 216–226. Association for Computational Linguistics, 2011.

[4] Haitao Mi, Liang Huang, and Qun Liu. Forest-based translation. In *ACL*, pages 192–199, 2008.

[5] Yang Liu, Qun Liu, and Shouxun Lin. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 609–616. Association for Computational Linguistics, 2006.

[6] Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 961–968. Association for Computational Linguistics, 2006.

[7] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics, 2001.

[8] Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

[9] Alfred V. Aho and Jeffrey D. Ullman. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56, 1969.

[10] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

[11] Richard Zens, Franz Josef Och, and Hermann Ney. Phrase-based statistical machine translation. In *KI 2002: Advances in Artificial Intelligence*, pages 18–32. Springer, 2002.

[12] Franz Josef Och, Christoph Tillmann, Hermann Ney, et al. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, 1999.

[13] Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302. Association for Computational Linguistics, 2002.

[14] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

[15] Philipp Koehn. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Machine translation: From real users to research*, pages 115–124. Springer, 2004.

[16] Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for Computational Linguistics, 2010.

[17] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[18] Liang Huang and Haitao Mi. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283. Association for Computational Linguistics, 2010.

[19] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[20] Andreas Stolcke. Srilm - an extensible language modeling toolkit. pages 901–904, 2002.

[21] Philipp Koehn Franz, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. pages 127–133, 2003.

[22] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics, 2003.

[23] Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS)*, pages 3–10. MIT Press, 2003.

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

[25] David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics, 2005.

[26] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics, 2006.

[27] David Chiang. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228, 2007.

[28] Hieu Hoang, Alexandra Birch, Chris Callison-burch, Richard Zens, Rwth Aachen, Alexandra Constantin, Marcello Federico, Nicola Bertoldi, Chris Dyer, Brooke Cowan, Wade Shen, Christine Moran, and Ondej Bojar. Moses: Open source toolkit for statistical machine translation. pages 177–180, 2007.

[29] 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*, StatMT '07, pages 40–47, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[30] Junhui Li, Zhaopeng Tu, Guodong Zhou, and Josef van Genabith. Head-driven hierarchical phrase-based translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 33–37, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[31] K. P. Chow E. Sumita I. Goto, B. Lu and B. K. Tsou. Overview of the patent machine translation task at the ntcir-10 workshop. In *Proc. of NTCIR-10*, 2012.

[32] David Guthrie and Mark Hepple. Storing the web in memory: Space efficient language models with constant time retrieval. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 262–272, Cambridge, MA, October 2010. Association for Computational Linguistics.

[33] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[34] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 380–388, New York, NY, USA, 2002. ACM.

[35] Moses Samson Charikar. Methods and apparatus for estimating similarity, December 2001. US Patent 7,158,961.

[36] Djamal Belazzougui, Fabiano C. Botelho, and Martin Dietzfelbinger. Hash, displace, and compress. In *ESA*, pages 682–693, 2009.

[37] Sebastiano Vigna. Broadword implementation of rank/select queries. In *Proceedings of the 7th international conference on Experimental algorithms*, WEA'08, pages 154–168, Berlin, Heidelberg, 2008. Springer-Verlag.

[38] Kimmo Fredriksson and Fedor Nikitin. Simple compression code supporting random access and fast string matching. In *Proceedings of the 6th international conference on Experimental algorithms*, WEA'07, pages 203–216, Berlin, Heidelberg, 2007. Springer-Verlag.

[39] Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

[40] Daniel M Cer, Marie-Catherine De Marneffe, Daniel Jurafsky, and Christopher D Manning. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *LREC*, 2010.

[41] Zhongyuan Zhu, Jun-ya Norimatsu, Toru Tanaka, Takashi Inui, and Mikio Yamamoto. Tsuku statistical machine translation system for the ntcir-10 patentmt task. In *Proceedings of NTCIR*, volume 10, 2013.