Hierarchical Chunk-to-String Translation

Yang Feng† Dongdong Zhang‡ Mu Li‡ Ming Zhou‡ Qun Liu⋆
† Department of Computer Science
University of Sheffield
Sheffield, UK
y.feng@shef.ac.uk

‡ Microsoft Research Asia
dozhang@microsoft.com
muli@microsoft.com
mingzhou@microsoft.com

⋆Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
liuqun@ict.ac.cn

Abstract

We present a hierarchical chunk-to-string translation model, which can be seen as a compromise between the hierarchical phrase-based model and the tree-to-string model, to combine the merits of the two models. With the help of shallow parsing, our model learns rules consisting of words and chunks and meanwhile introduces syntax cohesion. Under the weighed synchronous context-free grammar defined by these rules, our model searches for the best translation derivation and yields target translation simultaneously. Our experiments show that our model significantly outperforms the hierarchical phrase-based model and the tree-to-string model on English-Chinese Translation tasks.

1 Introduction

The hierarchical phrase-based model (Chiang, 2007) makes an advance of statistical machine translation by employing hierarchical phrases, which not only uses phrases to learn local translations but also uses hierarchical phrases to capture reorderings of words and subphrases which can cover a large scope. Besides, this model is formal syntax-based and does not need to specify the syntactic constituents of subphrases, so it can directly learn synchronous context-free grammars (SCFG) from a parallel text without relying on any linguistic annotations or assumptions, which makes it used conveniently and widely.

However, it is often desirable to consider syntactic constituents of subphrases, e.g. the hierarchical phrase

\[ X \rightarrow \langle X_1 \text{ for } X_2 , X_2 \text{ de } X_1 \rangle \]

can be applied to both of the following strings in Figure 1

“A request for a purchase of shares”
“filed for bankruptcy”,

and get the following translation, respectively

“goumai gufen de shenqing”
“pochan de shenqing”.

In the former, “A request” is a NP and this rule acts correctly while in the latter “filed” is a VP and this rule gives a wrong reordering. If we specify the first \( X \) on the right-hand side to NP, this kind of errors can be avoided.

The tree-to-string model (Liu et al., 2006; Huang et al., 2006) introduces linguistic syntax via source parse to direct word reordering, especially long-distance reordering. Furthermore, this model is formalised as Tree Substitution Grammars, so it observes syntactic cohesion. Syntactic cohesion means that the translation of a string covered by a subtree in a source parse tends to be continuous. Fox (2002) shows that translation between English and French satisfies cohesion in the majority cases. Many previous works show promising results with an assumption that syntactic cohesion explains almost all translation movement for some language pairs (Wu, 1997; Yamada and Knight, 2001; Eisner, 2003; Graehl and Knight, 2004; Quirk et al., 2005; Cherry, 2008; Feng et al., 2010).
But unfortunately, the tree-to-string model requires each node must be strictly matched during rule matching, which makes it strongly dependent on the relationship of tree nodes and their roles in the whole sentence. This will lead to data sparseness and being vulnerable to parse errors.

In this paper, we present a hierarchical chunk-to-string translation model to combine the merits of the two models. Instead of parse trees, our model introduces linguistic information in the form of chunks, so it does not need to care the internal structures and the roles in the main sentence of chunks. Based on shallow parsing results, it learns rules consisting of either words (terminals) or chunks (nonterminals), where adjacent chunks are packed into one nonterminal. It searches for the best derivation through the SCFG-motivated space defined by these rules and get target translation simultaneously. In some sense, our model can be seen as a compromise between the hierarchical phrase-based model and the tree-to-string model, specifically

- Compared with the hierarchical phrase-based model, it integrates linguistic syntax and satisfies syntactic cohesion.

- Compared with the tree-to-string model, it only needs to perform shallow parsing which introduces less parsing errors. Besides, our model allows a nonterminal in a rule to cover several chunks, which can alleviate data sparseness and the influence of parsing errors.

- we refine our hierarchical chunk-to-string model into two models: a loose model (Section 2.1) which is more similar to the hierarchical phrase-based model and a tight model (Section 2.2) which is more similar to the tree-to-string model.

The experiments show that on the 2008 NIST English-Chinese MT translation test set, both the loose model and the tight model outperform the hierarchical phrase-based model and the tree-to-string model, where the loose model has a better performance. While in terms of speed, the tight model runs faster and its speed ranking is between the tree-to-string model and the hierarchical phrase-based model.

\[ p(e|f) = \sum_c p(c|f)p(e|f, c) \] (1)
In our model, we employ rules containing non-terminals to handle long-distance reordering where boundary words play an important role. So for the subphrases which cover more than one chunk, we just maintain boundary chunks: we bundle adjacent chunks into one nonterminal and denote it as the first chunk tag immediately followed by “-” and next followed by the last chunk tag. Then, for the string pair `<filed for bankruptcy, shenqing pochan>`, we can get the rule

\[ r_1 : X \rightarrow \langle \text{VBN} \text{ NP}, \text{ NP} \rangle \]

while for the string pair `<A request for a purchase of shares, goumai gufen de shenqing>`., we can get

\[ r_2 : X \rightarrow \langle \text{NP} \text{ NP} \text{ NP} \rangle \]

The rule matching “A request for a purchase of shares was” will be

\[ r_3 : X \rightarrow \langle \text{NP} \text{ NP} \text{ VBD}, \text{ NP} \text{ NP} \text{ VBD} \rangle. \]

We can see that in contrast to the method of representing each chunk separately, this representation form can alleviate data sparseness and the influence of parsing errors.
Figure 3: The derivations of the sentence in Figure 1(a).

In these rules, the left-hand nonterminal symbol \( X \) can not match any nonterminal symbol on the right-hand side. So we need a set of rules such as

\[
\begin{align*}
NP &\rightarrow \langle X \rangle \\
NP-NP &\rightarrow \langle X, X \rangle
\end{align*}
\]

and so on, and set the probabilities of these rules to 1. To simplify the derivation, we discard this kind of rules and assume that \( X \) can match any nonterminal on the right-hand side.

Only with \( r_2 \) and \( r_3 \), we cannot produce any derivation of the whole sentence in Figure 1(a). In this case we need two special glue rules:

\[
\begin{align*}
r_4 : \quad S &\rightarrow \langle S, X \rangle \\
r_5 : \quad S &\rightarrow \langle X, X \rangle
\end{align*}
\]

Together with the following four lexical rules,

\[
\begin{align*}
r_6 : \quad X &\rightarrow \langle a \text{ request, shenqing} \rangle \\
r_7 : \quad X &\rightarrow \langle a \text{ purchase of shares, goumai gufen} \rangle \\
r_8 : \quad X &\rightarrow \langle \text{was, bei} \rangle \\
r_9 : \quad X &\rightarrow \langle \text{made, dijiao} \rangle
\end{align*}
\]

Figure 3(a) shows the derivation of the sentence in Figure 1(a).

2.2 A Tight Model

In the tight model, the right-hand side of each rule remains the same as the loose model, but the left-hand side nonterminal is not \( X \) but the corresponding chunk labels. If a rule covers more than one chunk, we just use the first and the last chunk labels to denote the left-hand side nonterminal. The rule set used in the tight model for the example in Figure 1(a) corresponding to that in the loose model becomes:

\[
\begin{align*}
r_2 : \quad \text{NP-NP} &\rightarrow \langle \text{NP for NP-NP}, \text{NP-NP de NP} \rangle \\
r_3 : \quad \text{NP-VBD} &\rightarrow \langle \text{NP-NP VBD}, \text{NP-NP VBD} \rangle \\
r_6 : \quad \text{NP} &\rightarrow \langle \text{a request, shenqing} \rangle \\
r_7 : \quad \text{NP-NP} &\rightarrow \langle \text{a purchase of shares, goumai gufen} \rangle \\
r_8 : \quad \text{VBD} &\rightarrow \langle \text{was, bei} \rangle \\
r_9 : \quad \text{VP} &\rightarrow \langle \text{made, dijiao} \rangle
\end{align*}
\]

During decoding, we first collect rules for each span. For a span which does not have any matching rule, if we do not construct default rules for it, there will be no derivation for the whole sentence, then we need to construct default rules for this kind of span by enumerating all possible binary segmentation of the chunks in this span. For the example in Figure 1(a), there is no rule matching the whole sentence,
so we need to construct default rules for it, which should be

\[ NP-\text{VP} \rightarrow \langle NP-VBD_1 \text{VP}_1, NP-VBD_2 \text{VP}_2 \rangle. \]
\[ NP-\text{VP} \rightarrow \langle NP-NP_1 VBD-\text{VP}_1, NP-NP_2 VBD-\text{VP}_2 \rangle. \]

and so on.

Figure 3(b) shows the derivation of the sentence in Figure 1(a).

3  Shallow Parsing

In a parse tree, a chunk is defined by a leaf node or an inner node whose children are all leaf nodes (See Figure 2 (a)). In our model, we identify chunks by traversing a parse tree in a breadth-first order. Once a node is recognized as a chunk, we skip its children. In this way, we can get a sole chunk sequence given a parse tree. Then we label each word with a label indicating whether the word starts a chunk (B-) or continues a chunk (I-). Figure 2(a) gives an example. In this method, we get the training data for shallow parsing from Penn Tree Bank.

We take shallow Parsing (chunking) as a sequence label task and employ Conditional Random Field (CRF)\(^1\) to train a chunker. CRF is a good choice for label tasks as it can avoid label bias and use more statistical correlated features. We employ the features described in Sha and Pereira (2003) for CRF. We do not introduce CRF-based chunker in this paper and more details can be got from Hammersley and Clifford (1971), Lafferty et al. (2001), Taskar et al. (2002), Sha and Pereira (2003).

4  Rule Extraction

In what follows, we introduce how to get the rule set. We learn rules from a corpus that first is bidirectionally word-aligned by the GIza++ toolkit (Och and Ney, 2000) and then is refined using a “final-and” strategy. We generate the rule set in two steps: first, we extract two sets of phrases, basic phrases and chunk-based phrases. Basic phrases are defined using the same heuristic as previous systems (Koehn et al., 2003; Och and Ney, 2004; Chiang, 2005). A chunk-based phrase is such a basic phrase that covers one or more chunks on the source side.

\[ \alpha, \beta \]
\[ \langle e_{j_1}^{f_1}, f_{j_1}^{e_1} \rangle \]

as follows:

1. A chunk-based phrase is a basic phrase;
2. \( c_{j_1} \) begins with “B-”;
3. \( f_{j_2} \) is the end word on the source side or \( c_{j_2+1} \) does not begins with “I-”.

Given a sentence pair \( \langle f, e, \sim \rangle \), we extract rules for the loose model as follows

1. If \( \langle f_{j_1}^{f_1}, e_{i_1}^{e_2} \rangle \) is a basic phrase, then we can have a rule
   \[ X \rightarrow \langle f_{j_1}^{f_1}, e_{i_1}^{e_2} \rangle \]
2. Assume \( X \rightarrow \langle \alpha, \beta \rangle \) is a rule with \( \alpha = \alpha_1 f_{j_1}^{f_2} \alpha_2 \) and \( \beta = \beta_1 e_{i_1}^{e_2} \beta_2 \), and \( \langle f_{j_1}^{f_2}, e_{i_1}^{e_2} \rangle \) is a chunk-based phrase with a chunk sequence \( Y_u \cdots Y_v \), then we have the following rule
   \[ X \rightarrow \langle \alpha_1 Y_u-Y_t \alpha_2, \beta_1 Y_u-Y_v \beta_2 \rangle. \]

We evaluate the distribution of these rules in the same way as Chiang (2007).

We extract rules for the tight model as follows

1. If \( \langle f_{j_1}^{f_2}, e_{i_1}^{e_2} \rangle \) is a chunk-based phrase with a chunk sequence \( Y_s \cdots Y_t \), then we can have a rule
   \[ Y_s-Y_t \rightarrow \langle f_{j_1}^{f_2}, e_{i_1}^{e_2} \rangle \]
2. Assume \( Y_s-Y_t \rightarrow \langle \alpha, \beta \rangle \) is a rule with \( \alpha = \alpha_1 f_{j_1}^{f_2} \alpha_2 \) and \( \beta = \beta_1 e_{i_1}^{e_2} \beta_2 \), and \( \langle f_{j_1}^{f_2}, e_{i_1}^{e_2} \rangle \) is a chunk-based phrase with a chunk sequence \( Y_u \cdots Y_v \), then we have the following rule
   \[ Y_s-Y_t \rightarrow \langle \alpha_1 Y_u-Y_t \alpha_2, \beta_1 Y_u-Y_v \beta_2 \rangle. \]

We evaluate the distribution of rules in the same way as Liu et al. (2006).

For the loose model, the nonterminals must be cohesive, while the whole rule can be noncohesive: if both ends of a rule are nonterminals, the whole rule is cohesive, otherwise, it may be noncohesive. In contrast, for the tight model, both the whole rule and the nonterminal are cohesive.

Even with the cohesion constraints, our model still generates a large number of rules, but not all
of the rules are useful for translation. So we follow the method described in Chiang (2007) to filter the rule set except that we allow two nonterminals to be adjacent.

5 Related Works

Watanabe et al. (2003) presented a chunk-to-string translation model where the decoder generates a translation by first translating the words in each chunk, then reordering the translation of chunks. Our model distinguishes from their model mainly in reordering model. Their model reorders chunks resorting to a distortion model while our model reorders chunks according to SCFG rules which retain the relative positions of chunks.

Nguyen et al. (2008) presented a tree-to-string phrase-based method which is based on SCFGs. This method generates SCFGs through syntactic transformation including a word-to-phrase tree transformation model and a phrase reordering model while our model learns SCFG-based rules from word-aligned bilingual corpus directly.

There are also some works aiming to introduce linguistic knowledge into the hierarchical phrase-based model. Marton and Resnik (2008) took the source parse tree into account and added soft constraints to hierarchical phrase-based model. Cherry (2008) used dependency tree to add syntactic cohesion. These methods work with the original SCFG defined by hierarchical phrase-based model and use linguistic knowledge to assist translation. Instead, our model works under the new defined SCFG with chunks.

Besides, some other researchers make efforts on the tree-to-string model by employing exponentially alternative parses to alleviate the drawback of 1-best parse. Mi et al. (2008) presented forest-based translation where the decoder translates a packed forest of exponentially many parses instead of 1-best parse. Liu and Liu (2010) proposed to parse and to translate jointly by taking tree-based translation as parsing. Given a source sentence, this decoder produces a parse tree on the source side and a translation on the target side simultaneously. Both the models perform in the unit of tree nodes rather than chunks.

6 Experiments

6.1 Data Preparation

Data for shallow parsing We got training data and test data for shallow parsing from the standard Penn Tree Bank (PTB) English parsing task by splitting the sections 02-21 on the Wall Street Journal Portion (Marcus et al., 1993) into two sets: the last 1000 sentences as the test set and the rest as the training set. We filtered the features whose frequency was lower than 3 and substituted ‘‘ and ‘‘ with ‘‘ to keep consistent with translation data. We used L2 algorithm to train CRF.

Data for Translation We used the NIST training set for Chinese-English translation tasks excluding the Hong Kong Law and Hong Kong Hansard\textsuperscript{2} as the training data, which contains 470K sentence pairs. For the training data set, we first performed word alignment in both directions using GIZA++ toolkit (Och and Ney, 2000) then refined the alignments using “final-and”. We trained a 5-gram language model with modified Kneser-Ney smoothing on the Xinhua portion of LDC Chinese Gigaword corpus. For the tree-to-string model, we parsed English sentences using Stanford parser and extracted rules using the GHKM algorithm (Galley et al., 2004).

We used our in-house English-Chinese data set as the development set and used the 2008 NIST English-Chinese MT test set (1859 sentences) as the test set. Our evaluation metric was BLEU-4 (Papineni et al., 2002) based on characters (as the target language is Chinese), which performed case-insensitive matching of n-grams up to $n = 4$ and used the shortest reference for the brevity penalty. We used the standard minimum error-rate training (Och, 2003) to tune the feature weights to maximize the BLEU score on the development set.

6.2 Shallow Parsing

The standard evaluation metrics for shallow parsing are precision P, recall R, and their harmonic mean F1 score, given by:

$$P = \frac{\text{number of exactly recognized chunks}}{\text{number of output chunks}}$$

$$R = \frac{\text{number of exactly recognized chunks}}{\text{number of reference chunks}}$$

The source side and target side are reversed.

\textsuperscript{2}The source side and target side are reversed.
Table 1: Shallow parsing result. The column Found gives the number of chunks recognized by CRF, the row All represents all types of chunks, and the row One represents the chunks that consist of one word.

For example, given a reference sequence B-NP I-NP I-NP B-VP I-VP B-VP, CRF outputs a sequence O-NP I-NP I-NP B-VP I-VP I-NP, then $P = 33.33\%$, $A = 66.67\%$.

Table 1 summaries the results of shallow parsing. For ‘‘ and ‘’ were substituted with ‘’, the performance was slightly influenced.

The F1 score of all chunks is 91.25% and the F1 score of One and NP, which in number account for about 90% of chunks, is 90.65% and 94.22% respectively. F score of NP chunking approaches 94.38% given in Sha and Pereira (2003).

### 6.3 Performance Comparison

We compared our loose decoder and tight decoder with our in-house hierarchical phrase-based decoder (Chiang, 2007) and the tree-to-string decoder (Liu et al., 2006). We set the same configuration for all the decoders as follows: stack size = 30, nbest size = 30. For the hierarchical chunk-based and phrase-based decoders, we set max rule length to 5. For the tree-to-string decoder, we set the configuration of rule extraction as: the height up to 3 and the number of leaf nodes up to 5.

We give the results in Table 2. From the results, we can see that both the loose and tight decoders outperform the baseline decoders and the improvement is significant using the sign-test of Collins et al. (2005) ($p < 0.01$). Specifically, the loose model has a better performance while the tight model has a faster speed.

Compared with the hierarchical phrase-based model, the loose model only imposes syntactic cohesion to nonterminals while the tight model imposes syntax cohesion to both rules and nonterminals which reduces search space, so it decoders faster. We can conclude that linguistic syntax can indeed improve the translation performance; syntactic cohesion for nonterminals can explain linguistic phenomena well; noncohesive rules are useful, too. The extra time consumption against hierarchical phrase-based system comes from shallow parsing.

By investigating the translation result, we find that our decoder does well in rule selection. For example, in the hierarchical phrase-based model, this kind of rules, such as

$$X \rightarrow \langle X \text{ of } X, * \rangle,$$

and so on, where * stands for the target component, are used with a loose restriction as long as the terminals are matched, while our models employ more stringent constraints on these rules by specifying the syntactic constituent of “X”. With chunk labels, our models can make different treatment for different situations.
Table 3: Influence of cohesion. The row cohesive represents the loose system where nonterminals satisfy cohesion, and the row noncohesive represents the modified version of the loose system where nonterminals can be noncohesive.

Table 4: The influence of the number of nonterminals. The column number lists the number of nonterminals used at most in a rule.

Compared with the tree-to-string model, the result indicates that the change of the source-side linguistic syntax from parses to chunks can improve translation performance. The reasons should be our model can reduce parse errors and it is enough to use chunks as the basic unit for machine translation. Although our decoders and tree-to-string decoder all run in linear-time with beam search, tree-to-string model runs faster for it searches through a smaller SCFG-motivated space.

6.4 Influence of Cohesion

We verify the influence of syntax cohesion via the loose model. The cohesive model imposes syntax cohesion on nonterminals to ensure the chunk is reordered as a whole. In this experiment, we introduce a noncohesive model by allowing a nonterminal to match part of a chunk. For example, in the noncohesive model, it is legal for a rule with the source side

"NP for NP-NP"

to match

"request for a purchase of shares"

in Figure 1 (a), where “request” is part of NP. As well, the rule with the source side

"NP for a NP-NP"

can match

"request for a purchase of shares".

In this way, we can ensure all the rules used in the cohesive system can be used in the noncohesive system. Besides cohesive rules, the noncohesive system can use noncohesive rules, too.

We give the results in Table 3. From the results, we can see that cohesion helps to reduce search space, so the cohesive system decodes faster. The noncohesive system decoder slower, as it employs more rules, but this does not bring any improvement of translation performance. As other researches said in their papers, syntax cohesion can explain linguistic phenomena well.

6.5 Influence of the number of nonterminals

We also tried to allow a rule to hold three nonterminals at most. We give the result in Table 4. The result shows that using three nonterminals does not bring a significant improvement of translation performance but quite more time consumption. So we only retain two nonterminals at most in a rule.

7 Conclusion

In this paper, we present a hierarchical chunk-to-string model for statistical machine translation which can be seen as a compromise of the hierarchical phrase-based model and the tree-to-string model. With the help of shallow parsing, our model learns rules consisting of either words or chunks and compresses adjacent chunks in a rule to a nonterminal, then it searches for the best derivation under the SCFG defined by these rules. Our model can combine the merits of both the models: employing linguistic syntax to direct decoding, being syntax cohesive and robust to parsing errors. We refine the hierarchical chunk-to-string model into two models: a loose model (more similar to the hierarchical phrase-based model) and a tight model (more similar to the tree-to-string model).

Our experiments show that our decoder can improve translation performance significantly over the hierarchical phrase-based decoder and the tree-to-string decoder. Besides, the loose model gives a better performance while the tight model gives a faster speed.
8 Acknowledgements

We would like to thank Trevor Cohn, Shujie Liu, Nan Duan, Lei Cui and Mo Yu for their help, and anonymous reviewers for their valuable comments and suggestions. This work was supported in part by EPSRC grant EP/I034750/1 and in part by High Technology R&D Program Project No. 2011AA01A207.

References

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In Proceedings of AMTA.