Tree Kernel-based SVM with Structured Syntactic Knowledge for BTG-based Phrase Reordering

Min Zhang **Haizhou Li**
Institute for Infocomm Research
1 Fusionopolis Way,#21-01 Connexis (South Tower)
Singapore 138632
{mzhang,hli}@i2r.a-star.edu.sg

Abstract

Structured syntactic knowledge is important for phrase reordering. This paper proposes using convolution tree kernel over source parse tree to model structured syntactic knowledge for BTG-based phrase reordering in the context of statistical machine translation. Our study reveals that the structured syntactic features over the source phrases are very effective for BTG constraint-based phrase reordering and those features can be well captured by the tree kernel. We further combine the structured features and other commonly-used linear features into a composite kernel. Experimental results on the NIST MT-2005 Chinese-English translation tasks show that our proposed phrase reordering model statistically significantly outperforms the baseline methods.

1 Introduction

Phrase-based method (Koehn et al., 2003; Och and Ney, 2004; Koehn et al., 2007) and syntax-based method (Wu, 1997; Yamada and Knight, 2001; Eisner, 2003; Chiang, 2005; Cowan et al., 2006; Marcu et al., 2006; Liu et al., 2007; Zhang et al., 2007c, 2008a, 2008b; Shen et al., 2008; Mi and Huang, 2008) represent the state-of-the-art technologies in statistical machine translation (SMT). As the two technologies are complementary in many ways, an interesting research topic is how to combine the strengths of the two methods. Many research efforts have been made to address this issue, which can be summarized into two ideas. One is to add syntax into phrase-based model while another one is to enhance syntax-based model to handle non-syntactic phrases. In this paper, we bring forward the first idea by studying the issue of how to utilize structured syntactic features for phrase reordering in a phrase-based SMT system with BTG (Bracketing Transduction Grammar) constraints (Wu, 1997).

Word and phrase reordering is a crucial component in a SMT system. In syntax-based method, word reordering is implicitly addressed by translation rules, thus the performance is subject to parsing errors to a large extent (zhang et al., 2007a) and the impact of syntax on reordering is difficult to single out (Li et al., 2007). In phrase-based method, local word reordering[1] can be effectively captured by phrase pairs directly while local phrase reordering is explicitly modeled by phrase reordering model and distortion model. Recently, many phrase reordering methods have been proposed, ranging from simple distance-based distortion model (Koehn et al., 2003; Och and Ney, 2004), flat reordering model (Wu, 1997; Zens et al., 2004), lexicalized reordering model (Tillmann, 2004; Kumar and Byrne, 2005), to hierarchical phrase-based model (Chiang, 2005; Setiawan et al., 2007) and classifier-based reordering model with linear features (Zens and Ney, 2006; Xiong et al., 2006; Zhang et al., 2007a; Xiong et al., 2008). However, one of the major limitations of these advances is the structured syntactic knowledge, which is important to global reordering (Li et al., 2007; Elming, 2008), has not been well exploited. This makes the phrase-based method particularly weak in handling global phrase reordering. From machine learning viewpoint (Vapnik, 1995), it is computationally infeasible to explicitly generate features involving structured information in many NLP applica-

[1] This paper follows the term convention of global reordering and local reordering of Li et al. (2007), between which the distinction is solely defined by reordering distance (whether beyond four source words) (Li et al., 2007).

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 698–707,
Singapore, 6-7 August 2009.

tions. For example, one cannot enumerate efficiently all the sub-tree features for a full parse tree. This would be the reason why structured features are not fully utilized in previous statistical feature-based phrase reordering model.

Thanks to the nice property of kernel-based machine learning method that can implicitly explore (structured) features in a high dimensional feature space (Vapnik, 1995), in this paper we propose using convolution tree kernel (Haussler, 1999; Collins and Duffy, 2001) to explore the structured syntactic knowledge for phrase reordering and further combine the tree kernel with other diverse linear features into a composite kernel to strengthen the model's predictive ability. Indeed, using tree kernel methods to mine structured knowledge has shown success in some NLP applications like parsing (Collins and Duffy, 2001), semantic role labeling (Moschitti, 2004; Zhang et al., 2007b), relation extraction (Zhang et al., 2006), pronoun resolution (Yang et al., 2006) and question classification (Zhang and Lee, 2003). However, to our knowledge, such technique still remains unexplored for phrase reordering.

In this paper, we look into the phrase reordering problem in two aspects: 1) how to model and optimize structured features, and 2) how to combine the structured features with other linear features and further integrate them into the log-linear model-based translation framework. Our study shows that: 1) the structured syntactic features are very useful and 2) our kernel-based model can well explore diverse knowledge, including previously-used linear features and the structured syntactic features, for phrase reordering. Our model displays one advantage over the previous work that it is able to utilize the structured syntactic features without the need for extensive feature engineering in decoding a parse tree into a set of linear syntactic features.

To have a more insightful evaluation, we design three experiments with three different evaluation metrics. Experimental results on the NIST MT-2005 Chinese-English translation tasks show that our method statistically significantly outperforms the baseline methods in term of the three different evaluation metrics.

The rest of the paper is organized as follows. Section 2 introduces the baseline method of BTG-based phrase translation method while section 3 discusses the proposed method in detail. The experimental results are reported and discussed in section 4. Finally, we conclude the paper in section 5.

2 Baseline System and Method

We use the MaxEnt-based BTG translation system (Xiong et al., 2006) as our baseline. It is a phrase-based SMT system with BTG reordering constraint. The system uses the BTG lexical translation rules ($A \rightarrow x/y$) to translate the source phrase x into target phrase y, and the BTG merging rules ($A \rightarrow [A, A] | < A, A >$) to combine two neighboring phrases with a straight or inverted order. In the translation model, the BTG lexical rules are weighted with several features, such as phrase translation, word penalty and language models, in a log-linear form. With the BTG constraint, the reordering model Ω is defined on the two neighboring phrases A^1 and A^2 and their order $o \in \{straight, inverted\}$ as follows:

$$\Omega = \mathrm{f}(o, A^1, A^2) \quad (1)$$

In the baseline system, a MaxEnt-based classifier with boundary words of the two neighboring phrases as features is used to model the merging/reordering order. The baseline MaxEnt-based reordering model is formulized as follows:

$$\Omega = p_\theta(o|A^1, A^2) = \frac{exp(\sum_i \theta_i h_i(o, A^1, A^2))}{\sum_o exp(\sum_i \theta_i h_i(o, A^1, A^2))} \quad (2)$$

where the functions $h_i(o, A^1, A^2) \in \{0,1\}$ are model feature functions using the boundary words of the two neighboring phrases as features, and θ_i are feature weights that are trained based on the MaxEnt-based criteria.

3 Tree Kernel-based Phrase Reordering Model

3.1 Kernel-based Classifier Solution to Phrase Reordering

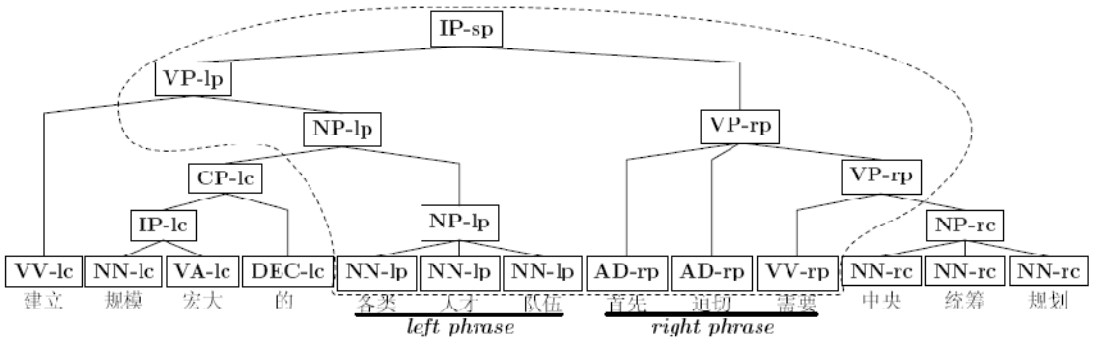
In this paper, phrase reordering is recast as a classification issue as done in previous work (Xiong et al., 2006 & 2008; Zhang et al., 2007a). In training, we use a machine learning algorithm training on the annotated phrase reordering instances that are automatically extracted from word-aligned, source sentence parsed training corpus, to learn a classifier. In testing (decoding), the learned classifier is applied to two adjacent source phrases to decide whether they should be merged (straight) or reordered (inverted) and what their probabilities are, and then these probabilities are used as one feature in the log-linear model in a phrase-based decoder.

In addition to the previously-used linear features, we are more interested in the value of structured syntax in phrase reordering and how to capture it using kernel methods. However, not

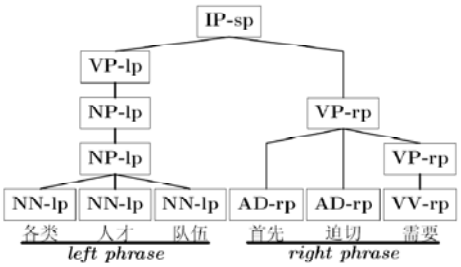
all classifiers are able to work with kernel methods. Only those dot-product-based classifiers can work with kernels by replacing the dot product with a kernel function, where the kernel function is able to directly calculate the similarity between two (structured) objects without enumerating them into linear feature vectors. In this paper, we select SVM as our classifier. In this section, we first define the structured syntactic features and introduce the commonly used linear features, and then discuss how to utilize these features by kernel methods together SVM for phrase reordering

3.2 Structured Syntactic Features

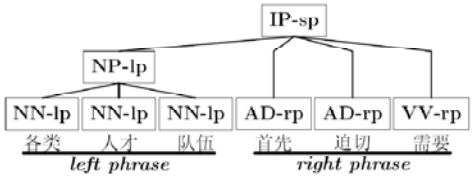
A reordering instance $x = \{A^1, A^2\}$ (see Eq.1) in this paper refers to two adjacent source phrases A^1 and A^2 to be translated. The structured syntactic feature spaces of a reordering instance are defined as the portion of a parse tree of the source sentence that at least covers the span of the reordering instance (i.e. the two neighboring phrases). The syntactic features are defined as all



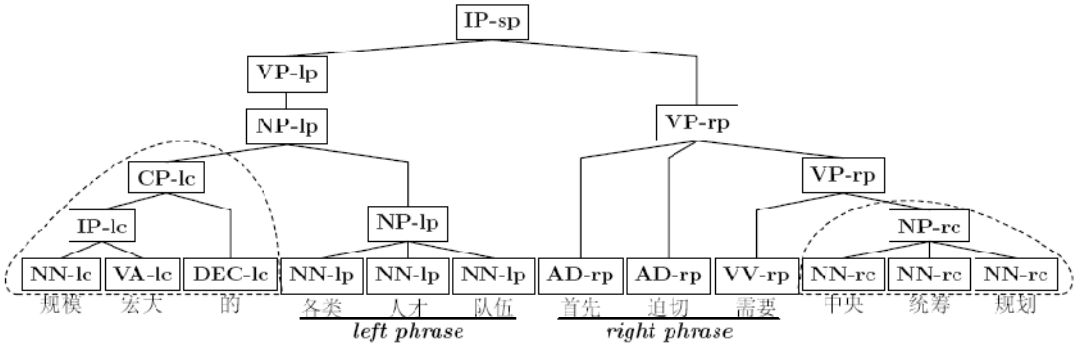
T_1) **Minimum Sub-Tree (MST)**



T_2) **Minimum Sub-Structure (MSS)**



T_4) **Chunking Tree (CT)**



T_3) **Context-sensitive Minimum Sub-Structure (CMSS)**

Figure 1. Different representations of structured syntactic features of a reordering instance in the example sentence excerpted from our training corpus "…建立/build 规模/scale 宏大/mighty 的/of 各类/various types 人才/qualified personnel 队伍/contingent 首先/above all 迫切/urgently 需要/necessary 中央/central authorities 统筹/overall 规划/planning…(To build a mighty contingent of qualified personnel of various types, it is necessary, above all, for the central authorities to make overall planning.) ", where "各类/various types 人才/qualified personnel 队伍/contingent (contingent of qualified personnel of various types)" is the 1st/left phrase and "首先/above all 迫切/urgent 需要/necessary (it is necessary, above all, …)" is the 2nd/right phrase. Note that different function tags are attached to the grammar tag of each internal node.

the possible subtrees in the structured feature spaces. We can see that the structured feature spaces and their features are encapsulated by a full parse tree of source sentences. Thus, it is critical to understand which portion of a parse tree (i.e. structured feature space) is the most effective to represent a reordering instance. Motivated by the work of (Zhang et al., 2006), we here examine four cases that contain different sub-structures as shown in Fig. 1.

(1) Minimum Sub-Tree (MST): the sub-tree rooted by the nearest common ancestor of the two phrases. This feature records the minimum sub-structure covering the two phrases and its left and right contexts as shown in Fig 1.T_1.

(2) Minimum Sub-Structure (MSS): the smallest common sub-structure covering the two phrases. It is enclosed by the shortest path linking the two phrases. Thus, its leaf nodes exactly consist of all the phrasal words.

(3) Context-sensitive Minimum Sub-Structure (CMSS): the **MSS** extending with the 1st left sibling node of the left phrase and the 1st right sibling node of the right phrase and their descendants. If sibling is unavailable, then we move to the parent of current node and repeat the same process until the sibling is available or the root of the **MST** is reached.

(4) Chunking Tree (CT): the base phrase list extracted from the **MSS**. We prune out all the internal structures of the **MSS** and only keep the root node and the base phrase list for generating the chunking tree.

Fig. 1 illustrates the different representations of an example reordering instance. T_1 is the **MST** for the example instance, where the sub-structure circled by a dotted line is the **MSS**, which is also shown in T_2 for clarity. We can see that the **MSS** is a subset of the **MST**. By T_2 we would like to evaluate whether the structured information is effective for phrase reordering while by comparing the system performance when using T_1 and T_2, we would like to evaluate whether the structured context information embedded in the **MST** is useful to phrase reordering. T_3 is the **CMSS**, where the two sub-structures circled by dotted lines are included as the context to T_2 and make T_3 limited context-sensitive. This is to evaluate whether the limited context information in the **CMSS** is helpful. By comparing the performance of T_1 and T_3, we would like to see whether the larger context in T_1 is a noisy feature. T_4 is the **CT**, where only the basic structured information is kept. By comparing the performance of T_2 and T_4, we would like to study whether the high-level structured syntactic features in T_2 are useful to phrase reordering.

After defining the four structured feature spaces, we further partition each feature space into five parts according to their functionalities. Because it only makes sense to evaluate two partitions of the same functionality between two reordering instances, the feature space partition leads to a more precise similarity calculation. As shown in Fig 1, all the internal nodes in each partition are labeled with a unique function tag in the following way:

- **Left Context (-lc):** nodes in this partition do not cover any phrase word and they are all in the left of the left phrase.
- **Right Context (-rc):** nodes in this partition do not cover any phrase word and they are all in the right of the right phrase.
- **Left Phrase (-lp):** nodes in this partition only cover the first phrase and/or its left context.
- **Right Phrase (-rp):** nodes in this partition only cover the second phrase and/or its right context.
- **Shared Part (-sp):** nodes in this partition at least cover both of the two phrases partially.

No lexical word is tagged since it is not a part of the structured features, and therefore not participating in the tree kernel computing.

3.3 Linear Features

In our study, we define the following lexicalized linear features which are easily to be extracted and integrated to our composite kernel:

- Leftmost and rightmost boundary words of the left and right source phrases
- Leftmost and rightmost boundary words of the left and right target phrases
- Internal words of the four phrases (excluding boundary words)
- Target language model (LM) score difference (monotone-inverted)

In total, we arrive at 13 features, including 8 boundary word features, 4 (kinds of) internal word features and 1 LM feature. The first 12 features have been proven useful (Xiong et al., 2006; Zhang et al., 2007a) to phrase reordering. LM score is certainly a strong evidence for modeling word orders and lexical selection. Although it is already used as a standalone feature in the log-linear model, we also would like to explicitly re-optimize it together with other reordering features in our reordering model.

3.4 Tree Kernel, Composite Kernel and Integrating into our Reordering Model

As discussed before, we use convolution tree kernel to capture the structured syntactic feature implicitly by directly computing similarity between the parse-tree representations of two reordering instances with explicitly enumerating all the features one by one. In convolution tree kernel (Collins and Duffy, 2001), a parse tree T is implicitly represented by a vector of integer counts of each sub-tree type (regardless of its ancestors):

$$\phi(T) = (\#\ subtree_1(T), \ldots, \#\ subtree_n(T))$$

where $\#\ subtree_i(T)$ is the occurrence number of the i^{th} sub-tree type ($subtree_i$) in T. Since the number of different sub-trees is exponential with the parse tree size, it is computationally infeasible to directly use the feature vector $\phi(T)$. To solve this computational issue, Collins and Duffy (2001) proposed the following parse tree kernel to calculate the dot product between the above high dimensional vectors implicitly.

$$\begin{aligned} K(T_1,T_2) &= <\phi(T_1),\phi(T_2)> \\ &= \sum_i \left(\left(\sum_{n_1\in N_1} I_{subtree_i}(n_1)\right)\cdot\left(\sum_{n_2\in N_2} I_{subtree_i}(n_2)\right)\right) \\ &= \sum_{n_1\in N_1}\sum_{n_2\in N_2}\Delta(n_1,n_2) \end{aligned}$$

where N_1 and N_2 are the sets of nodes in trees T_1 and T_2, respectively, and $I_{subtree_i}(n)$ is a function that is 1 iff the $subtree_i$ occurs with root at node n and zero otherwise, and $\Delta(n_1,n_2)$ is the number of the common *subtrees* rooted at n_1 and n_2, i.e.,

$$\Delta(n_1,n_2) = \sum_i I_{subtree_i}(n_1)\cdot I_{subtree_i}(n_2)$$

$\Delta(n_1,n_2)$ can be further computed efficiently by the following recursive rules:

Rule 1: if the productions (CFG rules) at n_1 and n_2 are different, $\Delta(n_1,n_2)=0$;

Rule 2: else if both n_1 and n_2 are pre-terminals (POS tags), $\Delta(n_1,n_2)=1\times\lambda$;

Rule 3: else,

$$\Delta(n_1,n_2) = \lambda\prod_{j=1}^{nc(n_1)}(1+\Delta(ch(n_1,j),ch(n_2,j)))$$

where $nc(n_1)$ is the child number of n_1, $ch(n,j)$ is the j^{th} child of node n and λ ($0<\lambda<1$) is the decay factor in order to make the kernel value less variable with respect to the *subtree* sizes. In addition, the recursive **Rule 3** holds because given two nodes with the same children, one can construct common sub-trees using these children and common sub-trees of further offspring. The time complexity for computing this kernel is $O(|N_1|\cdot|N_2|)$ and in practice in near to linear computational time without the need of enumerating all subtree features.

In our study, the linear feature-based similarity is simply calculated using dot-product. We then define the following composite kernel to combine the structured features-based and the linear features-based similarities:

$$K_c(x_1,x_2) = \alpha\cdot K_t(x_1,x_2) + (1-\alpha)\cdot K_l(x_1,x_2) \quad (3)$$

where K_t is the tree kernel over the structured features and K_l is the linear kernel (dot-product) over the linear features. The composite kernel K_c is a linear combination of the two individual kernels, where the coefficient α is set to its default value 0.3 as that in Moschitti (2004)'s implementation. The kernels return the similarities between two reordering instances based on their features used. Our basic assumption is, the more similar the two reordering instances of x_1 and x_2 are, the more chance they share the same order.

Now let us see how to integrate the kernel functions into SVM. The linear classifier learned by SVM is formulized as:

$$f(x) = \mathrm{sgn}(\textstyle\sum_i y_i a_i x\bullet x_i + b) \quad (4)$$

where a_i is the weight of a support vector x_i (i.e., a support reordering instance $x_i = \{A^1, A^2\}$in our study), y_i is its class label (1: $straight$ or -1: $inverted$ in our study) and b is the intercept of the hyperplane. An input reordering instance x is classified as positive (negative) if $f(x)>0$ ($f(x)<0$).

Based on the linear classifier, a kernelized SVM can be easily implemented by simply replacing the dot product $x*x_i$ in Eq (4) with a kernel function $K(x,x_i)$. Thus, the kernelized SVM classifier is formulated as:

$$f(x) = \mathrm{sgn}(\textstyle\sum_i y_i a_i K(x,x_i) + b) \quad (5)$$

where $K(x,x_i)$ is either $K_c(x,x_i)$, $K_t(x,x_i)$ or $K_l(x,x_i)$ in our study. Following Eq (1), our reordering model (implemented by the kernelized SVM) can be formulized as follows:

$$\begin{aligned} \Omega &= \mathrm{f}(o,A^1,A^2) = p_{svm}(o|x=\{A^1,A^2\}) \\ &= sgn(\textstyle\sum_i(y_i a_i K(x,x_i)+b)) \end{aligned} \quad (6)$$

A reordering instance x is classified as *straight* (or *inverted*) if $p_{svm}(o|x)>0$ (or $p_{svm}(o|x)<0$). Eq (6) and Eq (2) show the difference between our kernalized SVM-based reordering

model and the MaxEnt-based reordering model. The main difference between them lies in that our model is able to utilize structured syntactic features by kernalized SVM while the previous work can only use lexicalized word features by MaxEnt-based classifier.

Finally, because the return value of $p_{svm}(o|x)$ is a distance function rather than a probability, we use a sigmoid function to convert $p_{svm}(o|x)$ to a posterior probability as shown using the following to functions and apply it as one feature to the log-linear model in the decoding.

$$P(straight \mid x) = \frac{1}{1+e^{-p_{svm}(o|x)}} \quad \text{and}$$

$$P(inverted \mid x) = \frac{1}{1+e^{p_{svm}(o|x)}}$$

where *straight* represents a positive instance and *inverted* represents a negative instance.

4 Experiments and Discussion

4.1 Experimental Settings

Basic Settings: we evaluate our method on Chinese-English translation task. We use the FBIS corpus as training set, the NIST MT-2002 test set as development (dev) set and the NIST MT-2005 test set as test set. The Stanford parser (Klein and Manning, 2003) is used to parse Chinese sentences on the training, dev and test sets. GIZA++ (Och and Ney, 2004) and the heuristics "grow-diag-final-and" are used to generate *m-to-n* word alignments. The translation model is trained on the FBIS corpus and a 4-gram language model is trained on the Xinhua portion of the English Gigaword corpus using the SRILM Toolkits (Stolcke, 2002) with modified Kneser-Ney smoothing (Kenser and Ney, 1995). For the MER training (Och, 2003), we modify Koehn's MER trainer (Koehn, 2004) to train our system. For significance test, we use Zhang et al's implementation (Zhang et al, 2004).

Baseline Systems: we set three baseline systems: **B1**) Moses (Koehn et al., 2007) that uses lexicalized unigram reordering model to predict three orientations: monotone, swap and discontinuous; **B2**) MaxEnt-based reordering model with lexical boundary word features only (Xiong et al., 2006); **B3**) Linguistically annotated reordering model for BTG-based (LABTG) SMT (Xiong et al., 2008). For Moses, we used the default settings. We build a CKY-style decoder and integrate the corresponding reordering modelling methods into the decoder to implement the 2nd and the 3rd baseline systems and our system. Except reordering models, all the four systems use the same features in translation model, language model and distortion model as Moses in the log-linear framework. We tune the four systems using the strategies as discussed previously in this section.

Reordering Model Training: we extract all reordering instances from the *m-to-n* word-aligned training corpus. The reordering instances include the two source phrases, two target phrases, order label and its corresponding parse tree. We generate the boundary word features from the extracted reordering instances in the same way as discussed in Xiong et al. (2006) and use Zhang's MaxEnt Tools[2] to train a reordering model for the 2nd baseline system. Similarly, we use the algorithm 1 in Xiong et al. (2008) to extract features and use the same MaxEnt Tools to train a reordering model for the 3rd baseline system. Based on the extracted reordering instances, we generate the four structured features and the linear features, and then use the Tree Kernel Tools (Moschitti, 2004) to train our kernel-based reordering model (linear, tree and composite).

Experimental Design and Evaluation Metrics: we design three experiments and evaluate them using three metrics.

Classification-based: in the first experiment, we extract all reordering instances and their features from the dev and test sets, and then use the reordering models trained on the training set to classify (label) those instances extracted from the dev and test sets. In this way, we can isolate the reordering problem from the influence of others, such as translation model, pruning and decoding strategies, to better examine the reordering models' ability and to give analytical insights into the features. Classification Accuracy (CAcc), the percentage of the correctly labeled instances over all trials, is used as the evaluation metric.

Forced decoding[3]-based and normal decoding-based: the two experiments evaluate the reordering models through a real SMT system. The reordering model and the language model are the same in the two experiments. However, in forced decoding, we train two translation models, one using training data only while another using both

[2] http://homepages.inf.ed.ac.uk/s0450736/maxent.html

[3] A normal SMT decoder filters a translation model according to the source sentences, whereas in forced decoding, a translation model is filtered based on both source sentence and target references. In other words, in forced decoding, the decoder is forced to use those phrases whose translations are already in the references.

training, dev and test data. By forced decoding, we aim to isolate the reordering problem from those of OOV and lexical selections resulting from imperfect translation model in the context of a real SMT task. Besides the the case-sensitive BLEU-4 (Papineni et al., 2002) used in the two experiments, we design another evaluation metrics Reordering Accuracy (RAcc) for forced decoding evaluation. RAcc is the percentage of the adjacent word pairs with correct word order[4] over all words in one-best translation results. Similar to BLEU score, we also use the similar Brevity Penalty BP (Papineni et al., 2002) to penalize the short translations in computing RAcc. Finally, please note for the three evaluation metrics, the higher values represent better performance.

Feature Spaces	CAcc (%)	
	Dev	**Test**
Minimum Sub-Tree (**MST**)	89.87	89.92
Minimum Sub-Structure (**MSS**)	87.95	87.88
Context-Sensitive MSS (**CMSS**)	89.11	89.01
Chunking Tree (**CT**)	86.17	86.21
Linear Features (K_l)	90.79	90.46
K_l w/o using LM feature ($K_{l\text{-}LM}$)	84.24	84.06
Composite Kernel (K_c: **MST**+K_l)	92.98	92.67
MST w/o the 5 function tags	86.94	87.03
All are *straight* (*monotonic*)	78.92	78.67

Table 1: Performance of our methods on the dev and test sets with different feature combinations

4.2 Experimental Results

Classification of Instances: Table 1 reports the performance of our defined four structured features, linear feature and the composite kernel. The results are summarized as follows.

The last row reports the performance without using any reordering features. We just suppose that all the translations are monotonic, no reordering happens. The CAccs of 78.92% and 78.67% serve as the bottom line in our study. Compared with the bottom line, the tree kernels over the 4 structured features are very effective for phrase reordering since only structured information is used in the tree kernel[5].

The **CT**s performs the worst among the 4 structured features. This suggests that the middle and high-level structures beyond base phrases are very useful for phrase reordering. The **MSS**s show lower performance than the **CMSS**s and the **MST**s achieve the best performance. This clearly indicates that the structured context information is useful for phrase reordering. For this reason, the subsequent discussions are focused on the **MST**s, unless otherwise specified. The **MSS**s without using the 5 function tags perform much worse than the original ones. This suggests that the partitions of the structured feature spaces are very helpful, which can effectively avoid the undesired matching between partitions of different functionalities. Comparison of K_l and $K_{l\text{-}LM}$ shows the LM plays an important role in phrase reordering. The composite kernel (K_c) performs much better than the two individual kernels. This suggests that the structured and linear features are complementary and the composite kernel can well integrate them for phrase reordering.

Methods	CAcc (%)	
	Dev	**Test**
Minimum Sub-Tree (**MST**)	89.87	89.92
Linear Features (K_l)	90.79	90.46
Composite Kernel (K_c: **MST**+K_l)	92.98	92.67
MaxEnt+boundary word (**B2**)	88.33	86.97
MaxEnt+linguistic features (**B3_1**)	84.83	83.92
MaxEnt+LABTG (**B3: B2+ B3_1**)	88.82	88.18

Table 2: Performance comparison of different methods

Table 2 compares the performance of the baseline methods with ours. Comparison between **B3_1** and **MST** clearly demonstrates that the structured syntactic features are much more effective than the linear syntactic features that are manually extracted via heuristics. It also suggests that the tree kernel can well capture the structured features implicitly. K_l outperforms **B2**. This is mainly due to the contribution of LM features. **B2** (MaxEnt-based) significantly outperforms $K_{l\text{-}LM}$ in Table 1 (SVM-based). This suggests that phrase reordering may not be a good linearly binary-separable task if only boundary word features are used. Our composite kernel (K_c) significantly outperforms LABTG (**B3**). This mainly

[4] An adjacent word pair $w_i w_{i+1}$ in a translation have correct word order if and only if w_i appears before w_{i+1} in translation references. Note than the two words may not be adjacent in the references even if they have correct word order.

[5] The tree kernel algorithm only compares internal structures. It does not concern any lexical leaf nodes.

attributes to the contributions of structured syntactic features, LM and the tree kernel.

Forced Decoding: Table 3 compares the performance of our composite kernel with that of the LABTG (Baseline 3) in forced decoding. As discussed before, here we try two translation models.

The composite kernel outperforms the LABTG in all test cases. This further validates the effectiveness of the kernel methods in phrase reordering. There are still around 30% words reordered incorrectly even if we use the translation model trained on both training, dev and test sets. This reveals the limitations of current SMT modeling methods and suggests interesting future work in this area. The source language OOV[6] rate in forced decoding (13.6%) is much higher that in normal decoding (6.22%, see table 4). This is mainly due to the fact that the phrase table in forced decoding is filtered out based on both source and target languages while in normal decoding it is based on source language only. As a result, more phrases are filtered out in the forced decoding. There is 1.4% OOV even if the translation model is trained on the test set. This is due to the incorrect word alignment, large-span word alignment and different English tokenization strategies used in BLEU-scoring tool and ours.

Methods	Test Set (%)		
	RAcc	OOV	BLEU
Composite Kernel (K_c)	51.03	13.6	38.56
+translation model on Training, dev and test	72.67	1.41	62.87
MaxEnt+LABTG (**B3**)	48.96	13.6	37.32
+translation model on training, dev and test	71.45	1.41	62.14

Table 3: Performance comparison of forced decoding

Methods	Test Set	
	BLEU(%)	OOV(%)
Composite Kernel (K_c)	27.65	6.26
Moses (**B1**)	25.71	6.17
MaxEnt+boundary word(**B2**)	25.99	6.22
MaxEnt+LABTG (**B3**)	26.63	6.22

Table 4: Performance comparison

[6] OOV means a source words has no any English translation according to the translation model. OOV rate is the percentage of the number of OOV words over all the source words.

Normal Decoding/Translation: Table 4 reports the translation performance of our system and the three baseline systems.

Moses (**B1**) and the MaxEnt-based boundary word model (**B2**) achieve comparable performance. This means the lexicalized orientation-based reordering model in Moses performs similarly to the boundary word-based reordering model since the two models are both lexical word-based. However, theoretically, the MaxEnt-based model may suffer less from data sparseness issue since it does not depends on internal phrasal words and uses MaxEnt to optimize feature weights while the orientation-based model uses relative frequency of the entire phrases to compute the posterior probabilities. s. The MaxEnt-based LABTG model significantly outperforms (p<0.05) the MaxEnt-based boundary word model and the lexicalized orientation-based reordering model. This indicates that the linearly linguistically syntactic information is a useful feature to phrase reordering.

Our composite kernel-based model significantly outperforms (p<0.01) the three baseline methods. This again proves that the structured syntactic features are much more effective than the linear syntactic features for phrase reordering and the tree kernel method can well capture the informative structured features. The four methods show very slight difference in OOV rates. This is mainly due to the difference in implementation detail, such as different OOV penalties and other pruning thresholds.

5 Conclusion and Future Work

Structured syntactic knowledge is very useful to phrase reordering. This paper provides insights into how the structured feature can be used for phrase reordering. In previous work, the structured features are selected manually by heuristics and represented by a linear feature vector. This may largely compromise the contribution of the structured features to phrase reordering. Thanks to the nice properties of kernel-based learning method and SVM classifier, we propose leveraging on the kernelized SVM learning algorithm to address the problem. Specifically, we propose using convolution tree kernel to capture the structured features and design a composite kernel to combine the structured features and other linear features for phrase reordering. The tree kernel is able to directly take the structured reordering instances as inputs and compute their similarities without enumerating them into a set of liner

features. In addition, we also study how to find the optimal structured feature space and how to partition the structured feature spaces according to their functionalities. Finally, we evaluate our method on the NIST MT-2005 Chinese-English translation tasks. To provide insights into the model, we design three kinds of experiments together with three different evaluation metrics. Experimental results show that the structured features are very effective and our composite kernel can well capture both the structured and the linear features without the need for extensive feature engineering. It also shows that our method significantly outperforms the baseline methods.

The tree kernel-based phrase reordering method is not only applicable to adjacent phrases. It is able to work with any long phrase pairs with gap of any length in-between. We will study this case in the near future. We would also like to use one individual tree kernel for one partition in a structured feature space. In doing so, we are able to give different weights to different partitions according to their functionalities and contributions. Note that these weights can be automatically tuned and optimized easily against a dev set.

References

David Chiang. 2005. *A hierarchical phrase-based model for SMT*. ACL-05. 263-270.

Michael Collins and N. Duffy. 2001. *Convolution Kernels for Natural Language*. NIPS-2001.

M. R. Costa-jussà and J.A.R. Fonollosa. 2006. *Statistical Machine Reordering*. EMNLP-06. 70-76.

Brooke Cowan, Ivona Kucerova and Michael Collins. 2006. *A discriminative model for tree-to-tree translation*. EMNLP-06. 232-241.

Jason Eisner. 2003. *Learning non-isomorphic tree mappings for MT*. ACL-03 (companion volume).

Jakob Elming. 2008. *Syntactic Reordering Integrated with Phrase-Based SMT*. COLING-08. 209-216.

Michel Galley and Christopher D. Manning. 2008. *A Simple and Effective Hierarchical Phrase Reordering Model*. EMNLP-08. 848-856.

David Haussler. 1999. *Convolution Kernels on Discrete Structures*. TR UCS-CRL-99-10.

T. Joachims. 1998. *Text Categorization with SVM: learning with many relevant features*. ECML-98.

Dan Klein and Christopher D. Manning. 2003. *Accurate Unlexicalized Parsing*. ACL-03. 423-430.

Reinhard Kenser and Hermann Ney. 1995. *Improved backing-off for M-gram language modeling*. ICASSP-95, 181-184

Philipp Koehn, F. Och and D. Marcu. 2003. *Statistical phrase-based translation*. HLT-NAACL-03.

Philipp Koehn, H. Hoang, A. Birch, C. C.-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin and E. Herbst. 2007. *Moses: Open Source Toolkit for SMT*. ACL-07 (poster). 77-180.

Shankar Kumar and William Byrne. 2005. *Local Phrase Reordering Models for Statistical Machine Translation*. HLT-EMNLP-2005. 161-168.

Chi-Ho Li, Dongdong Zhang, Mu Li, Ming Zhou, Minghui Li and Yi Guan. 2007. *A Probabilistic Approach to Syntax-based Reordering for Statistical Machine Translation*. ACL-07. 720-727.

Yang Liu, Yun Huang, Qun Liu and Shouxun Lin. 2007. *Forest-to-String Statistical Translation Rules*. ACL-07. 704-711.

Daniel Marcu, W. Wang, A. Echihabi and K. Knight. 2006. *SPMT: SMT with Syntactified Target Language Phrases*. EMNLP-06. 44-52.

Haitao Mi and Liang Huang. 2008. *Forest-based Translation Rule Extraction*. EMNLP-08. 206-214.

Alessandro Moschitti. 2004. *A Study on Convolution Kernels for Shallow Semantic Parsing*. ACL-04.

Masaaki Nagata, Kuniko Saito, Kazuhide Yamamoto and Kazuteru Ohashi. 2006. *A Clustered Global Phrase Reordering Model for Statistical Machine Translation*. COLING-ACL-06. 713-720.

Franz J. Och and Hermann Ney. 2002. *Discriminative training and maximum entropy models for statistical machine translation*. ACL-02. 295-302.

Franz J. Och. 2003. *Minimum error rate training in statistical machine translation*. ACL-03. 160-167.

Franz J. Och and H. Ney. 2003. *A Systematic Comparison of Various Statistical Alignment Methods*. Computational Linguistics, 29(1):20-51.

Franz J. Och and H. Ney. 2004. *The alignment template approach to statistical machine translation*. Computational Linguistics, 30(4):417-449.

Kishore Papineni, S. Roukos, T. and W. Zhu. 2002. *BLEU: a method for automatic evaluation of machine translation*. ACL-02. 311-318.

Hendra Setiawan, Min-Yen Kan and Haizhou Li. 2007. *Ordering Phrases with Function Words*. ACL-07. 712-719.

Libin Shen, Jinxi Xu and Ralph Weischedel. 2008. *A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model*. ACL-HLT-08. 577-585.

Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit*. ICSLP-02. 901-904.

Christoph Tillmann. 2004. *A Unigram Orientation Model for Statistical Machine Translation*. HLT-NAACL-04 (short paper).

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Chao Wang, M. Collins and P. Koehn. 2007. *Chinese Syntactic Reordering for Statistical Machine Translation*. EMNLP-CONLL-07. 734-745.

Dekai Wu. 1997. *Stochastic inversion transduction grammars and bilingual parsing of parallel corpora*. Computational Linguistics, 23(3):377-403.

Fei Xia and Michael McCord. 2004. *Improving a Statistical MT System with Automatically Learned Rewrite Patterns*. COLING-04.

Deyi Xiong, Qun Liu and Shouxun Lin. 2006. *Maximum Entropy Based Phrase Reordering Model for SMT*. COLING-ACL-06. 521−528.

Deyi Xiong, Min Zhang, Aiti Aw and Haizhou Li. 2008. *A Linguistically Annotated Reordering Model for BTG-based Statistical Machine Translation*. ACL-HLT-08 (short paper). 149-152.

Kenji Yamada and K. Knight. 2001. *A syntax-based statistical translation model*. ACL-01. 523-530.

Xiaofeng Yang, Jian Su and Chew Lim Tan. 2006. *Kernel-Based Pronoun Resolution with Structured Syntactic Knowledge*. COLING-ACL-06. 41-48.

Richard Zens, H. Ney, T. Watanabe and E. Sumita. 2004. *Reordering Constraints for Phrase-Based Statistical Machine Translation*. COLING-04.

Richard Zens and Hermann Ney. 2006. Discriminative Reordering Models for Statistical Machine Translation. WSMT-2006.

Dell Zhang and W. Lee. 2003. *Question classification using support vector machines*. SIGIR-03.

Min Zhang, Jie Zhang, Jian Su and GuoDong Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with Both Flat and Structured Features*. COLING-ACL-06. 825-832.

Dongdong Zhang, M. Li, C.H. Li and M. Zhou. 2007a. *Phrase Reordering Model Integrating Syntactic Knowledge for SMT*. EMNLP-CONLL-07. 533-540.

Min Zhang, W. Che, A. Aw, C. Tan, G. Zhou, T. Liu and S. Li. 2007b. *A Grammar-driven Convolution Tree Kernel for Semantic Role Classification*. ACL-07. 200-207.

Min Zhang, Hongfei Jiang, Ai Ti Aw, Jun Sun, Sheng Li and Chew Lim Tan. 2007c. *A Tree-to-Tree Alignment-based Model for Statistical Machine Translation*.MT-Summit-07. 535-542

Min Zhang, Hongfei Jiang, Ai Ti Aw, Haizhou Li, Chew Lim Tan and Chew Lim Tan and Sheng Li. 2008a. *A Tree Sequence Alignment-based Tree-to-Tree Translation Model*. ACL-HLT-08. 559-567.

Min Zhang, Hongfei Jiang, Haizhou Li, Aiti Aw, Sheng Li. 2008b. *Grammar Comparison Study for Translational Equivalence Modeling and Statistical Machine Translation*. COLING-08. 1097-1104

Ying Zhang, Stephan Vogel and Alex Waibel. 2004. *Interpreting BLEU/NIST scores: How much improvement do we need to have a better system?* LREC-04. 2051-2054.