Language Modeling for Syntax-Based Machine Translation Using Tree Substitution Grammars: A Case Study on Chinese-English Translation

TONG XIAO, JINGBO ZHU, and MUHUA ZHU, Northeastern University

The poor grammatical output of Machine Translation (MT) systems appeals syntax-based approaches within language modeling. However, previous studies showed that syntax-based language modeling using (Context-Free) Treebank Grammars was not very helpful in improving BLEU scores for Chinese-English machine translation. In this article we further study this issue in the context of Chinese-English syntax-based Statistical Machine Translation (SMT) where Synchronous Tree Substitution Grammars (STSGs) are utilized to model the translation process. In particular, we develop a Tree Substitution Grammar-based language model for syntax-based MT, and present three methods to efficiently integrate the proposed language model into MT decoding. In addition, we design a simple and effective method to adapt syntax-based language models for MT tasks. We demonstrate that the proposed methods are able to benefit a state-of-the-art syntax-based MT system. On the NIST Chinese-English MT evaluation corpora, we finally achieve an improvement of 0.6 BLEU points over the baseline.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Machine translation*

General Terms: Algorithms, Experimentation, Languages

Additional Key Words and Phrases: Machine translation, tree substitution grammar, syntax-based language model

ACM Reference Format:
Xiao, T., Zhu, J., and Zhu, M. 2011. Language modeling for syntax-based machine translation using tree substitution grammars: A case study on Chinese-English translation. ACM Trans. Asian Lang. Inform. Process. 10, 4, Article 18 (December 2011), 29 pages.
DOI = 10.1145/2025384.2025386 http://doi.acm.org/10.1145/2025384.2025386

1. INTRODUCTION

In recent years many statistical approaches have been developed in Chinese-English Machine Translation (MT), including phrase-based approaches [Koehn et al. 2003] hierarchical phrase-based approaches [Chiang 2005], and syntax-based approaches [Ding and Palmer 2005; Galley et al. 2004; Liu et al. 2006; Yamada and Knight 2001; Zhang et al. 2008]. The translation accuracy on Chinese-English translation has

This work was supported in part by the National Science Foundation of China (60873091; 61073140), Specialized Research Fund for the Doctoral Program of Higher Education (20100042110031), the Fundamental Research Funds for the Central Universities and Natural Science Foundation of Liaoning Province of China.
Authors' address: T. Xiao, J. Zhu (corresponding author), and M. Zhu, The Key Laboratory of Medical Image Computing (Ministry of Education), The Natural Language Processing Lab, Northeastern University, Shenyang, China, 110004; email: xiaotong@mail.neu.edu.cn, zhujingbo@mail.neu.edu.cn, zhumuhua@gmail.com.

DOI 10.1145/2025384.2025386 http://doi.acm.org/10.1145/2025384.2025386

continuously improved in recent MT competitions such as the NIST/DARPA MT Evaluation[1] and the IWSLT Evaluation Campaign[2].

Although Chinese-English translation quality has improved significantly, pervasive problems remain. One of them is that the output of most machine translation systems is very poor from the grammatical standpoint. To address this issue, it is natural to explore Syntax-based Language Modeling (SBLM) methods that make full use of the syntactic information on the target-language side to overcome the shortcomings of traditional *n*-gram language modeling. However, several research groups have reported that directly using (Context-Free) Treebank Grammar-based parsers[3] as Language Models (LMs) is not very helpful in improving BLEU scores for Chinese-English MT [Charniak et al. 2001; Och et al. 2004; Post and Gildea 2008]. For example, Charniak et al. [2001] used a syntactic parser to rescore a translation forest generated by a syntax-based MT system [Yamada and Knight 2001]. However, they obtained a lower BLEU score in spite of more grammatical outputs produced by using their parser. Cherry and Quirk [2008] used a parser with a Markovized, parent-annotated Treebank Grammar to distinguish grammatical from ungrammatical text. They found that such a syntax-based language model did not even appear very useful in classifying good and bad sentences.

While previous efforts on using parsers as language models did not show promising BLEU improvement, syntax-based language modeling is still one of the juiciest problems in MT due to its undoubted advantages over *n*-gram language modeling, such as the good abilities to model head/modifier dependencies [Knight and Koehn 2009]. In this work we further discuss this issue in the context of syntax-based MT. As Synchronous Tree Substitution Grammars (STSG) have been successfully used in syntax-based MT models in recent years[4], we focus on studying and experimenting with a state-of-the-art MT system based on the STSG model proposed in Galley et al. [2004] and Galley et al. [2006].

We argue that traditional syntax-based language models [Charniak et al. 2001; Och et al. 2004] do not work well in STSG-based MT due to two major reasons.

— First, treebank grammar-based parsers are somewhat unsuitable for detecting the well-formedness of translations produced by STSG-based MT systems. In general, traditional syntax-based language model is a syntactic parser that is parameterized and trained on all standard height-one subtrees defined in Treebanks, while the output of STSG-based MT systems is in principle an English syntax tree generated by applying a series of TSG rules involving larger tree fragments (i.e., subtrees of height ≥ 1). Though TSGs are recognized to be weakly equivalent to Context-Free Grammars (CFGs) which are classic Treebank Grammars, their formalisms are very different from each other. For example, TSGs are able to capture very complicated dependencies, while CFGs have limited capabilities in dealing with dependencies within a relatively large scope. Straightforwardly using the model score produced by CFG-based parsers, therefore, might be inappropriate for evaluating the well-formedness of target-syntax trees generated by STSG-based MT systems.

[1]See http://www.itl.nist.gov/iad/mig//tests/mt/.
[2]See http://iwslt2010.fbk.eu/.
[3]According to [Charniak 1996], a treebank grammar is *a context-free grammar created by reading the production rules directly from hand-parsed sentences in a Treebank*. In some cases it is also called *Context-Free Treebank Grammar*. Note that the concept of grammar is not explicitly used in some parsers, such as the Charniak Parser and the Collins Parser. Here we use term *treebank grammar* to emphasize that the parser is modeled and parameterized on height-one subtrees in treebanks, like standard CFGs.
[4]For example, synchronous tree substitution grammar-based model is the basis of one of the most successful systems in NIST 2009 MT evaluation.

— Second, the syntax-based language model is trained on the treebank data with correct English and manual annotation (e.g. the Penn Treebank), but is tested on out-of-domain data with full of incorrect English generated by MT systems. As the underlying probability distribution on Treebank is quite different from that in the MT environment, it is not expected that the language model could perform well when applied to non-treebank translation domain. A more severe problem is that a traditional syntax-based language model is essentially a syntactic parser whose design is not for the purpose of language modeling and grammaticality detection. Since the output of MT system is generally far from perfect English, even is incorrect English, straightforwardly using a general parser to re-score MT outputs might result in unexpected translation results in some cases [Koehn 2010].

In this article we investigate methods to address the preceding issues.

— We develop a language model, called *TSG-based language model*, for syntax-based MT using tree substitution grammars and demonstrate that it is able to improve the BLEU score of a state-of-the-art Chinese-English MT system.
— We present three methods to efficiently integrate the TSG-based language model into MT decoding.
— We present a simple and effective method to adapt the TSG-based language model for syntax-based MT, aiming at guiding the language model training using the data that is relatively more “preferred” by MT systems.

The rest of this article is structured as follows. In Section 2, we briefly describe the baseline MT system used in this study. We then present a TSG-based language model for syntax-based MT in Section 3. In Section 4, we describe three methods for efficient language model integration into decoding. Furthermore, in Section 5 we design a method to adapt syntax-based language models for MT. In Section 6, we present the experimental evaluation of the proposed methods. After reviewing the related work, we finally conclude this article with a summary and outlook of further work.

2. THE SYNTAX-BASED SYSTEM

2.1 Synchronous Tree Substitution Grammar

In this work, the baseline system is a Chinese-English Statistical Machine Translation (SMT) system based on the string-to-tree model proposed in Galley et al. [2004] and Galley et al. [2006]. This type of translation model can be regarded as a special instance of the framework of synchronous tree substitution grammars and tree transducers [Knight and Graehl 2005], and provides a formalism that can synchronously generate a pair of source-language sentence and target-language syntax tree. Typically, a translation rule used in this model maps a source-language CFG rule (representing a string) into a target-language TSG rule (representing a tree structure). The following example is a sample string-to-tree rule

$$\mathbf{S} \rightarrow <\mathbf{NP_1 PP_2}\ \text{表示 满意},\ \mathbf{NP_1}\ \text{VP(VBZ(was) VP(VBN(satisfied)}\mathbf{PP_2}\text{))}>$$

which explains the mapping from the CFG rule “**S** → **NP PP** 表示 满意” to the TSG rule “**S** → **NP** VP(VBZ(was) VP(VBN(satisfied) **PP**))”. As the CFG rule can be regarded as a “simplified” TSG rule, the string-to-tree rule can also be viewed as a rule from TSG to TSG. In this work, we use terms *STSG rule* and *STSG* to name the string-to-tree rule and the grammar in the string-to-tree model, though they are a little different from the definitions in the traditional STSG formalism where tree structures are required to be in both source-language and target-language sides [Chiang and Knight 2006].

According to Chiang and Knight's [2006] definition, an STSG rule consists of three parts, a left-hand side and two right-hand sides (call them the *source-side* and the *target-side*). For example, for the above STSG rule, the left-hand side **S** is the *root* for the synchronous production. The source-side "$\mathbf{NP_1}$ $\mathbf{PP_2}$ 表示 满意" and the target-side "$\mathbf{NP_1}$ VP(VBZ(was) VP(VBN(satisfied) $\mathbf{PP_2}$))" represent the source tree fragment and the target tree fragment, respectively[5]. The subscript numbers indicate one-to-one alignments that link frontier non-terminals on the source-side to frontier non-terminals on the target-side. For example, $\mathbf{NP_1}$ links with $\mathbf{NP_1}$, $\mathbf{PP_2}$ links with $\mathbf{PP_2}$ and so on. The non-terminal-labeled frontier nodes are generally called substitution nodes that can be rewritten recursively during synchronous parsing. In the rewriting operation (or substitution operation), an aligned pair of substitution nodes is rewritten with an STSG rule, guaranteeing that the substitution nodes must match the root of the STSG rule with which they are rewritten.

In a sense, the target-side of an STSG rule is a subtree in the target-language, having words (terminals) and variables (non-terminals) at leaves. Thus we can draw a target-language TSG rule from the target-side of an STSG rule. For example, from the above STSG rule, we can induce a target-language TSG rule like this:

$$\mathbf{S} \rightarrow \mathbf{NP}\ \mathrm{VP(VBZ(was)\ VP(VBN(satisfied)}\ \mathbf{PP}))$$

Obviously, it is a grammar rule of monolingual TSG, which implies that the synchronous generation can also explain the generation of the target-language syntax tree. In other words, the string-to-tree model generates the target-tree by recursively applying the target-language TSG rules, and meanwhile generates the source-language string correspondingly using the source-side of STSG rules.

2.2 Rule Extraction

Typically, the string-to-tree model is trained on the word-aligned bilingual text whose target-side has been parsed using an English syntactic parser. To obtain basic STSG rules, the minimal GHKM extraction method proposed in Galley et al. [2004] is utilized. The basic idea of GHKM extraction is to compute the set of the minimally-sized rules that can explain the mappings between source-language string and target-language tree while respecting the alignment and reordering between the two languages. For example, from the string-tree pair shown at the top of Figure 1, we can extract minimal GHKM rules $r_1 - r_6$. In addition to the GHKM extraction, we compose two or more minimal rules having shared states to form larger rules [Galley et al. 2006; Marcu et al. 2006]. For example, rule r_7 in Figure 1 is generated by composing rule r_2 and rule r_4.

2.3 Decoding

When we translate from a Chinese sentence f to an English sentence e using the extracted rules, the system builds a Chinese tree from f, and meanwhile generates an English tree for e. Our primary translation model is based on the joint probability $P(\pi_e, \pi_f)$ where π_e and π_f denote the trees on the English side and the Chinese side, respectively. Tuple (π_e, π_f) is generally called synchronous tree, and can be explained by a derivation of STSG rules used in decoding. The translation task can be described as: given a Chinese sentence f, we find an optimal English translation e^* by taking

[5]The source-side can be regarded as a height-one tree fragment, though no explicit source-language syntax is used in the model.

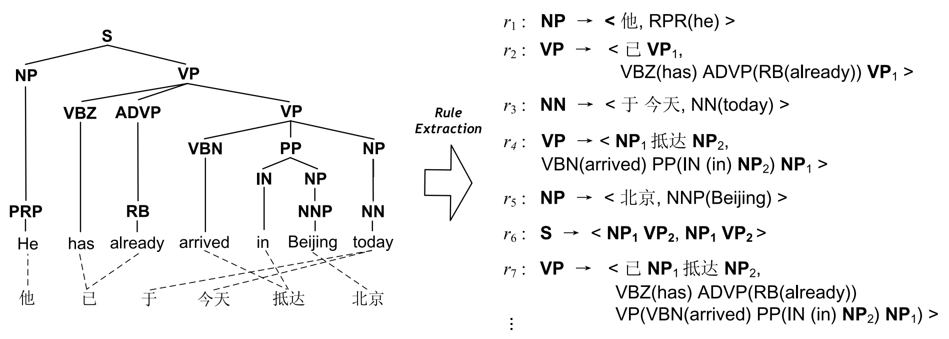

S
NP
VP
VBZ
ADVP
VP
VBN
PP
NP
IN
NP
NNP
NN
PRP
RB
He has already arrived in Beijing today
他 已 于 今天 抵达 北京
Rule Extraction
r1 : NP → < 他, RPR(he) >
r2 : VP → < 已 VP1, VBZ(has) ADVP(RB(already)) VP1 >
r3 : NN → < 于 今天, NN(today) >
r4 : VP → < NP1 抵达 NP2, VBN(arrived) PP(IN (in) NP2) NP1 >
r5 : NP → < 北京, NNP(Beijing) >
r6 : S → < NP1 VP2, NP1 VP2 >
r7 : VP → < 已 NP1 抵达 NP2, VBZ(has) ADVP(RB(already)) VP(VBN(arrived) PP(IN (in) NP2) NP1) >
⋮

Fig. 1. A word-aligned string-tree pair and sample STSG rules extracted.

the yield of the highest probability tree π_e. This process is formulized as the following equations.

$$\begin{aligned} e^* &= \arg\max_e P(e|f) \\ &= \arg\max_e P(e, f) \\ &\equiv \text{yield}(\arg\max_{\pi_e} P(\pi_e, \pi_f)) \end{aligned} \tag{1}$$

where yield(π_e) is a function that returns the yield of a given tree π_e. To define $P(\pi_e, \pi_f)$, a (log-)linear model [Och and Ney 2002] is used to combine a group of features.

$$\arg\max_{\pi_e} P(\pi_e, \pi_f) = \arg\max_{\pi_e} \sum_{m=1}^{M} \lambda_m h_m(\pi_e, \pi_f) \tag{2}$$

where $\{h_m(\pi_e, \pi_f)|m = 1, ..., M\}$ is a set of features, and λ_m is the feature weight corresponding to the m-th feature. In our baseline system, the feature design is mainly based on a state-of-the-art system described in Marcu et al. [2006]. There are 14 features in total, including bidirectional lexical and phrase-based translation probabilities (four features), a syntactic feature (root normalized conditional probability), a 5-gram language model, target word penalty, several binary features (lexicalized rule; low-frequency rule; composed rule; name, number, and date translation rules), and a word-deletion feature. All feature weights are optimized using Minimal Error Rate Training [Och 2003].

It is worth noting that in our system the syntactic feature is designed to model the process of synchronous tree generation. Given a synchronous tree, it computes the probability of how likely this synchronous tree is generated. Similar to the tree probability in Context-Free Grammar (CFG) parsing, the probability of a synchronous tree is the product of all synchronous grammar rule probabilities. In this work the rule probability is defined as the root normalized conditional probability which is estimated on the extracted synchronous grammar. We choose this type of rule probability because it has been successfully used in several syntax-based systems [Galley et al. 2006; Marcu et al. 2006; Mi and Huang 2008].

We use a CKY-style decoder with cube pruning [Huang and Chiang 2007] and beam search to decode new Chinese sentences. For efficient integration of n-gram language model into decoding, rules containing more than two variables are binarized using the synchronous binarization method [Zhang et al. 2006].

3. SYNTAX-BASED LANGUAGE MODELING FOR SYNTAX-BASED MT

3.1 Motivation

As described in Section 2.3, the objective of the baseline model is to find the best English syntax tree for a given Chinese sentence. It is important, therefore, that the model can concentrate more on well-formed English trees, and thus generate more grammatical translation outputs.

An obvious solution to this issue is that we use syntactic parsers as language models—call them syntax-based language models—to assess how likely the most possible English tree is well-formed. That is, for a tree-structured translation π_e, a parser is used to calculate the tree probability $\mathrm{P}(\pi_e)$ (or model score given by the parser). $\mathrm{P}(\pi_e)$ is then used to estimate the well-formedness of π_e. In this article we call the calculation of $\mathrm{P}(\pi_e)$ target-tree scoring, or simply tree scoring for short. Applying syntax-based language models to the syntax-based system provides several advantages. First, syntactic parsing is the process of analyzing a text to determine its grammatical structure with respect to a given (more or less) formal grammar[6]. Comparing to *n*-gram language modeling, it is more suitable for judging the grammaticality in translations, such as detecting subject-verb relations. Moreover, as the design of the parser is generally linguistically motivated, it is believed that parsers can offer reliable tree scoring for syntax-based MT. Second, parsing is a well-studied task in the Natural Language Processing (NLP) community [Jurafsky and Martin 2008]. Many statistical (or partly statistical) models are available to automatically learn tree-scoring functions from a corpus of data that has already been syntactically annotated (by hand or automatically). Third, the output of syntax-based MT is already in scorable tree form. It means that we do not need to re-parse the translation candidates or recover their underlying syntactic structures during decoding. Instead, each tree-structured translation candidate (i.e., π_e) is just rescored by the syntax-based language model.

In the rest parts of this section, we first describe the traditional method of syntax-based language modeling using context-free treebank grammars, and then present a new language modeling method based on tree substitution grammars.

3.2 Language Modeling Using Context-Free Treebank Grammars

To compute $\mathrm{P}(\pi_e)$, a simple and straightforward method is to use the tree scoring functions of existing parsers that have been trained on treebanks. Typically, these parsers are based on the model derived from parse-annotated training corpora or treebanks, and are parameterized using standard height-one subtrees (or CFG rules) defined in the training data. In this article, we refer to this type of parser as treebank grammar-based parser, or simply treebank parser for short.

To date, treebank parsing models have been intensively investigated. A number of well-developed parsers are available for this study. Among them, we choose two state-of-the-art ones as a starting point to experiment with syntax-based language modeling.

PCFG Model

Probabilistic Context-Free Grammar (PCFG) is one of the simplest grammar formalisms for statistical parsing. Under PCFG models, the probability of a syntax tree is simply the product of rule probabilities.

$$\mathrm{P}_{cfg}(\pi_e) = \prod_{r \in \pi_e} \mathrm{P}_{cfg}(r) \tag{3}$$

[6]See http://en.wikipedia.org/wiki/Parsing.

where r is a PCFG rule (i.e., a subtree of height one) induced from Treebanks, and $\mathrm{P}_{cfg}(r)$ is the rule probability of r. Typically, $\mathrm{P}_{cfg}(r)$ is estimated on the training data using Maximum Likelihood Estimation (MLE). Because $\mathrm{P}_{cfg}(\pi_e)$ has a bias towards trees with fewer rule applications, we use the normalized PCFG probability for syntax-based language modeling in this work.

$$\mathrm{P}(\pi_e) \equiv \mathrm{P}_{cfg}(\pi_e)^{1/N}, \tag{4}$$

where N is the number of rules used in generating π_e.

Lexicalized PCFG Model (Collins Model 2):

In addition to (non-lexicalized) PCFG models, lexicalized models [Charniak 1997; Collins 1999] are also very good choices for syntax-based language modeling. In this type of parsing model, each rule is specialized for one or more lexical items, which offers some nice advantages over non-lexicalized models. For example, lexicalized models are able to express syntactic preferences that are sensitive to lexical words, while non-lexicalized models cannot. Due to these advantages, parsers based on lexicalized models have achieved state-of-the-art parsing performance over the past few years. In this work we also employ a state-of-the-art lexicalized model – Collins Model 2 – to experiment with syntax-based language modeling.

In the Collins parsing model 2, the generation of a syntax tree is decomposed into many sub-steps, using reasonable independence assumptions. More formally, for a lexicalized grammar rule, the generation process can be described as:

(1) Choose a head H with probability $P_H(H|P, w_h)$, where H is the label of generated head child, P is the label of parent node, and w_h is the head word.
(2) Choose left and right subcategorization frames, $subcat_l$ and $subcat_r$, with probabilities $P_l\,(subcat_l|P, H, w_h)$ and $P_r\,(subcat_r|P, H, w_h)$.
(3) Generate all left and right modifiers with probabilities $P_l(Li, li|\Theta l(i))$ and $P_r(Rj, rj|\Theta r(j))$. Here Li(or Rj) denotes the label of the i-th left (or the j-th right) modifier, li (or rj) denotes the head word of Li(or Rj), $\Theta l(i)$ (or $\Theta r(j)$) denotes the parameters which the probabilities condition on. $\Theta l(i)$ and $\Theta r(j)$ generally contains many parameters, such as the distance between modifier and head word.

All parameters described above can be learned from treebanks. Given a trained model, the probability of a syntax tree π_e is defined to be the product of probabilities of all these sub-steps.

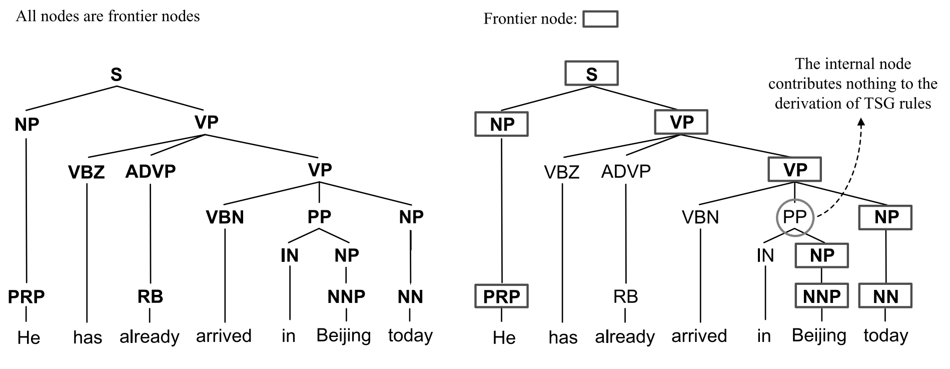
$$\begin{aligned} P_{lex}(\pi_e) \equiv & \prod_{\text{Rule}} P_H(H|P, w_h) \\ & \times P_l(subcat_l|P, H, w_h) \times P_r(subcat_r|P, H, w_h) \\ & \times (\textstyle\prod_i P_l(Li, li|\Theta l(i))) \\ & \times (\textstyle\prod_j P_r(Rj, rj|\Theta r(j))) \end{aligned} \tag{5}$$

Like PCFG-based language modeling, the tree probability is finally normalized with the number of grammar rules used in generating π_e (i.e., N):

$$\mathrm{P}(\pi_e) \equiv \mathrm{P}_{lex}(\pi_e)^{1/N} \tag{6}$$

3.3 Language Modeling Using Tree Substitution Grammars

As described in Section 2, the string-to-tree decoder seeks the best derivation by parsing the input sentence with an STSG. This process is very similar to that used in Data Oriented Parsing (DOP) Models [Bonnema 2002], where a deviation is generated by applying a series of rules (or tree fragments) with height ≥ 1. It means that the


All nodes are frontier nodes
Frontier node:
The internal node contributes nothing to the derivation of TSG rules
S
NP
VP
VBZ
ADVP
VP
VBN
PP
NP
IN
NP
PRP
RB
NNP
NN
He
has
already
arrived
in
Beijing
today

(a) The tree generated using height-one sub-trees (CFG rules) (b) The tree generated using TSG rules

Fig. 2. A syntax tree generated by using CFG rules (a) or TSG rules (b). In (a) all nodes are frontier nodes, while in (b) the nodes in red boxes are frontier nodes. As can be seen from (b), the internal node PP is useless for the generation of the TSG tree.

target-tree generated by our MT system is actually produced using the target-side of STSG rules instead of CFG rules (or height-one TSG rules) defined in a Treebank. For example, in our baseline model, more than 81% of the rules are STSG rules whose target-side varies in height from 2 to 10, while only 19% of the rules are STSG rules with a height of one. In the set of rules used during decoding, 71% and 29% of the rules are of height ≥ 2 and height = 1, respectively. In the set of rules used in generating the final (1-best) translations, the two numbers are 67% and 33%. These observations indicate that the syntax-based MT system is "dominated" by STSG rules of height ≥ 2 rather than height-one STSG rules.

To illustrate this point more clearly, Figure 2 compares the same syntax tree generated using different grammars (a CFG and a TSG). We can see that even to generate the same tree, the CFG and TSG rule applications are quite different. It is a risk that CFGs may be not as good as language models for syntax-based MT as often thought, because the grammar formalisms and parameterizations are quite different between within language modeling and MT. For example, the internal nodes of TSG subtrees contribute nothing to STSG-based MT, but affect CFG-based language modeling. A natural question that arises is whether the syntax-based language models can be improved using other alternatives such as TSGs rather than CFGs.

3.3.1 Basic Method. In this section, we explore solutions to this issue, using TSGs within language modeling. Instead of exploiting other sophisticated models such as DOP models, we still use the PCFG and lexicalized PCFG (Collins Model 2) models for language modeling. The basic idea is very simple. We train parsing models on the target-side of the bilingual data using TSGs, and then employ the resulting parsers as language models for syntax-based MT. This method provides two obvious advantages over the CFG-based language modeling. First, as target-language TSGs are involved, the language model is more suitable for scoring the translations produced by STSG-based MT systems. Second, the training of language models fits MT better, since language models are optimized on the same data as that used in MT, rather than on the limited out-of-domain Treebank data (compared to MT). To illustrate the difference between various language modeling methods, Figure 3 shows a comparison.

The only issue we need to address here is how to train these parsing models (including both the PCFG and the lexicalized PCFG models) on the target-side of bilingual training data using TSGs. To deal with it, we use a simple solution inspired by Post and Gildea [2009] which preprocesses the corpus in three ways before conducting

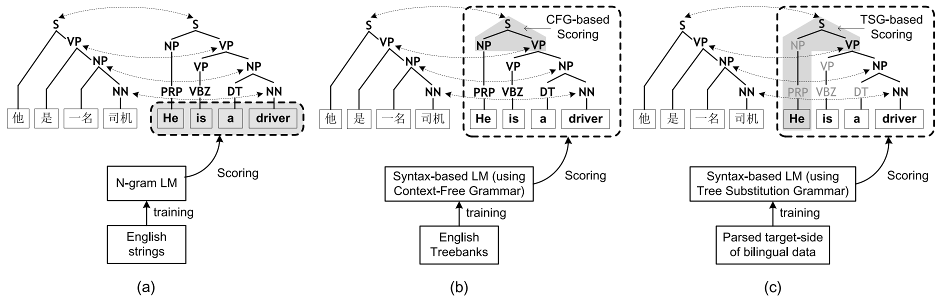

他 是 一名 司机
He is a driver
N-gram LM
Scoring
training
English strings
Syntax-based LM (using Context-Free Grammar)
English Treebanks
CFG-based Scoring
Syntax-based LM (using Tree Substitution Grammar)
Parsed target-side of bilingual data
TSG-based Scoring
(a)
(b)
(c)

Fig. 3. Comparison of different language modeling methods. (a), (b), and (c) show the cases of traditional *n*-gram language modeling, CFG-based language modeling and TSG-based language modeling, respectively. The doted arrows represent the connections between the equivalent non-terminals in the synchronous trees.

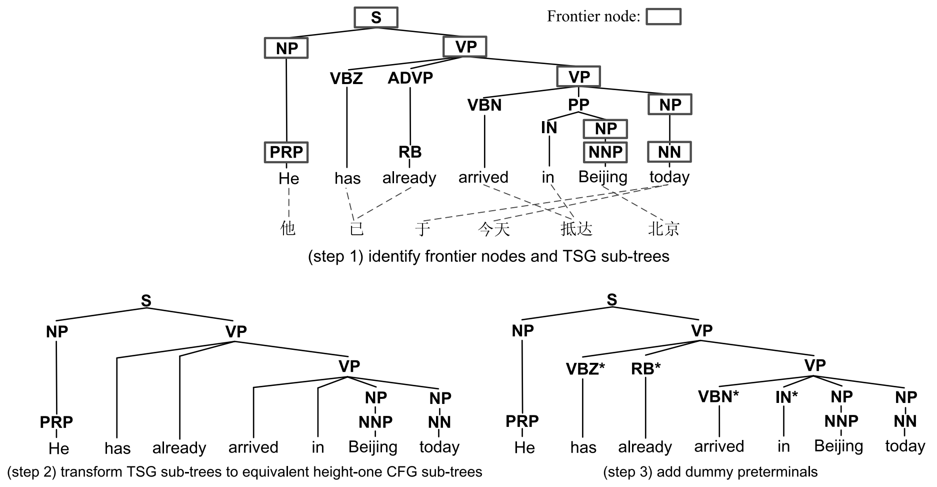

Frontier node:
He has already arrived in Beijing today
他 已 于 今天 抵达 北京
(step 1) identify frontier nodes and TSG sub-trees
(step 2) transform TSG sub-trees to equivalent height-one CFG sub-trees
(step 3) add dummy preterminals

Fig. 4. Illustration of the three-step preprocessing method for TSG-based language model training [Post and Gildea 2009].

the parser training. First, we compute frontier nodes in syntax trees (with respect to word alignment) and then identify all minimal TSG subtrees corresponding to minimal GHKM rules [Galley et al. 2004]. Second, we flatten these TSG subtrees to equivalent height-one CFG trees. As the internal nodes of these TSG subtrees have no use for MT, it will not affect the result under our TSG-based language models. Finally, we introduce dummy preterminals to the words whose tags are removed during subtree-flattening. Figure 4 illustrates this method with a tiny example.

After the three-step preprocessing, each English tree π in the bilingual training corpus is transformed into a simpler form $\Omega(\pi)$. Consequently, we can collect the statistics for both the PCFG and Collins parsing models from $\Omega(\pi)$[7]. These models are then used as language models in our syntax-based system. Also, the STSG rules can be extracted using $\Omega(\pi)$ instead of π because they are equivalent for both frontier identification and TSG rule extraction. This procedure is illustrated in Figure 5(b), with a comparison with the traditional method (Figure 5(a)) described in Section 3.2.

[7]We also modify the head-finding rules in Collins parser for selecting the correct heads when some necessary interior nodes are absent.

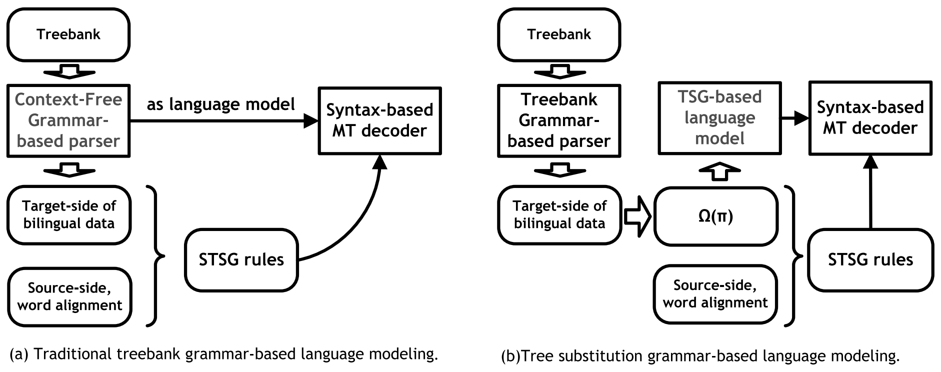

Treebank
Context-Free Grammar-based parser
as language model
Syntax-based MT decoder
Target-side of bilingual data
Source-side, word alignment
STSG rules
(a) Traditional treebank grammar-based language modeling.
Treebank
Treebank Grammar-based parser
TSG-based language model
Syntax-based MT decoder
Target-side of bilingual data
Ω(π)
Source-side, word alignment
STSG rules
(b)Tree substitution grammar-based language modeling.

Fig. 5. Illustration of context-free treebank grammar-based language modeling (a) and tree substitution grammar-based language modeling (b).

Finally, for each target-tree π_e generated by the MT system, we score it using $P(\Omega(\pi_e))$, where $P(\cdot)$ is the syntax-based language model that is trained using TSGs[8]. The definition of $P(\cdot)$ is the same as that described in Section 3.2. So we omit the detailed description about it in this section.

3.3.2 Beyond Minimal Rules. In general, there are many ways to partition a parse tree into TSG rule applications. As described above, in the training of TSG-based LM, we choose the simplest partition that is constrained by the minimal GHKM extraction. By using such a partition, each resulting TSG rule corresponds to the target-side of a minimal GHKM rule, and is equivalent to a CFG rule in the transformed tree. Although this method can alleviate the data sparseness in transformation, it may limit the capability of the proposed model to only capture the minimal GHKM rules. Obviously, our model can be enhanced by using more TSG partitions and estimating parameters from the TSG rules in the transformed tree instead of the CFG rules.

Motivated by this idea, we extend the approach described above to handle more TSG rules beyond the minimal rules. However, as in DOP [Bonnema 2002], there is generally an exponentially large number of TSG partitions for a parse tree. This makes the training of our model very complex. To ease the problem, in this work we only consider some critical TSGs instead of all TSGs. Our method is inspired from the rule composing [Galley et al. 2006] and the SPMT rule extraction [Marcu et al. 2006] in MT, which are two standard methods to improve the limited scope of the GHKM extraction. The idea is very simple: our model uses all TSG rules in the transformed tree that are consistent with the rule composing and the SPMT extraction. That is, instead of training parameters from the CFG rules in the transformed tree structure, we also estimate the TSG rules that are obtained by composing a number of CFG rules (or minimal rules) or SPMT extraction. For example, in Step 3 of Figure 4, three CFG rules (or minimal rules) can be identified, as follows.

$$\begin{aligned} \textbf{PRP} &\rightarrow \text{He} \\ \textbf{NP} &\rightarrow \textbf{PRP} \\ \textbf{S} &\rightarrow \textbf{NP VP} \end{aligned}$$

[8]Because π_e is generated using target-language TSG rules (i.e., target-side of STSG rules), the frontier nodes of π_e can be easily identified without word alignment.

These rules can be further composed into a larger rule which is then involved in the TSG-based LM.

$$\mathbf{S} \rightarrow \text{NP(PRP(He))}\ \mathbf{VP}$$

Note that the above method only considers a small sub-set of all TSG partitions—TSG rules that are consistent with the translation rule extraction used in MT. Fortunately, the completeness of the resulting TSG rules can be guaranteed, because the target-side of any translation rule is estimated in our TSG-based LM. In other words, when an STSG rule is applied during MT decoding, we can always find its target-side (also a TSG rule) and the corresponding probability in the TSG-based LM. This property makes the TSG "compact" enough to encode the probabilities of all potential target-trees generated by the MT decoder. In addition, this method has a practical advantage that the TSG rules can be obtained by simply reusing the rule extraction component of our MT system. It means that, in the training of the TSG-based LM, no additional computation is required to enumerate the large number of TSG rules, and the training is nearly as fast as the original version of our method (as described in Section 3.3.1).

3.3.3 Language Modeling Without Removing Internal Nodes. In our proposed method, we remove all internal nodes that are not identified as the frontier nodes in the original trees. This treatment has two advantages: first, it can alleviate the data sparseness problem caused by large-sized and low-frequency TSG rules; and second, removing internal nodes can ease the modeling and training of the TSG-based language model, since they are not really used in our pipeline of string-to-tree SMT[9]. However, this method has a potential risk that TSGs are less discriminative due to the lack of structure difference in TSG rules. For example, for the rule r_7 in Figure 1, the internal node **PP** is removed during the tree transformation as described in Section 3.3.1. As a result, the tree structure **PP**(**IN NP**) is ignored in rule extraction. It means that the resulting rules cannot distinguish the syntactic structures covering **IN NP** and thus somewhat less discriminative for both STSG and LM. Although no studies have shown that the removal of internal nodes could result in decreased performance of MT, it is still worth a discussion on the issue. To this end, we enhance our method by recovering the internal structures which are ignored in the tree transformation (Step 2 in Figure 4) for language modeling. The improved method keeps all the information in the original parse trees. We give an empirical comparison of the method described in Section 3.3.1 and the improved method described herein, which will be shown in the experimental evaluation (Section 6).

4. INCORPORATING SYNTAX-BASED LANGUAGE MODELS INTO MT DECODING

So far we have studied different methods for syntax-based language modeling. A natural issue that arises is how to incorporate syntax-based language models into MT decoding. So in this section we present three methods for syntax-based language model integration.

[9]Depending what decoding strategy is used, the internal nodes have different usage in different SMT paradigms [Liu and Liu 2010]. For example, if we treat decoding as parsing [Galley et al. 2006; Marcu et al. 2006], the internal nodes of tree fragments in translation rules contribute nothing to the generation of the translations; if we treat decoding as tree parsing [Eisner 2003; Liu et al. 2006], the internal nodes are usually used in matching the translation rules onto the input source-language parse tree generated by the syntactic parser.

4.1 Syntax-Based Language Models as Integrated Features

As our MT decoder is basically a parser that works in a bottom-up fashion, the syntax-based language model can be formulized as a feature that is fit directly into the decoding and integrated into the translation model. Hence we can treat the syntax-based language model as a new feature in the (log-)linear model, and re-formulize Equation (2) as:

$$\arg\max_{\pi_e} \Pr(\pi_e, \pi_f) = \arg\max_{\pi_e} \left(\sum\nolimits_{m=1}^{M} \lambda_m h_m(\pi_e, \pi_f) + \lambda_{SBLM} h_{SBLM}(\pi_e) \right) \qquad (7)$$

where $\sum_{m=1}^{M} \lambda_m h_m(\pi_e, \pi_f)$ is the baseline model, and $h_{SBLM}(\pi_e) = \log(\mathrm{P}(\pi_e))$is the syntax-based LM feature. During decoding, the model score of translation hypothesis is calculated with the baseline model and the additional syntax-based LM together.

4.2 Re-ranking

Compared to treating syntax-based language model as an integrated feature, a simpler method is to re-rank the *n*-best output of the MT system using a rich feature set containing the syntax-based language model. This method can be regarded as an instance of multi-pass decoding [Jurafsky and Martin 2008]. In the first stage, the baseline system returns an *n*-best list of translation candidates for a given Chinese sentence. Then, in the second stage, the *n*-best translation candidates are re-sorted using a more sophisticated model. In this work, our re-ranking feature set consists of the baseline features and the syntax-based language model. The re-ranking system is based on the linear model described in Equation (7) and implemented by reusing the decoder and MERT components of our baseline system.

4.3 Local Re-ranking of Translation Hypotheses

The two methods described above are both rational solutions to the problem, but each of them has its own advantages and disadvantages. On one hand, the traditional re-ranking method is simple to implement and has been successfully utilized in the related studies [Charniak 2001; Koehn et al. 2003; Och et al. 2004]. However, traditional re-ranking is very sensitive to the quality of *n*-best list (or translation forest). In most cases, the desired translation is not included in the *n*-best list due to the early stage pruning. Thus, it might be late to apply syntax-based language model in the post-processing stage. On the other hand, as the calculation of $\mathrm{P}(\pi_e)$ is very time-consuming, the integration of syntax-based language model in MT decoding is generally not very efficient.

In this section we instead propose a new method, *local re-ranking* (LR), to take the best use of both traditional re-ranking and feature integration methods. The basic idea is that we locally re-rank the *n*-best translation hypotheses that share the same properties during decoding. Take CKY-style decoding for instance. For each source span, we first generate the *n*-best translation hypotheses using the baseline model, and then re-rank them using a sophisticated model that contains the syntax-based language model.

Let $f = f_1 ... f_m$ be a Chinese sentence, and $Q[p,q]$ be the set of translation hypotheses for the source word sequence spanning from position p to q. The below pseudocode formulizes the local re-ranking algorithm within the framework of CKY-style decoding.

The major part of this algorithm is the procedure of standard CKY-style decoding (lines 2–7). Line 6 indicates the generation of translation hypotheses from the consecutive source spans [p, k] and [k+1, q], and line 7 indicates the update of the *n*-best list of the source span [p, q]. In line 9, the *n*-best hypotheses are re-ranked using the same model and features as those described in the traditional re-ranking system

(Section 4.2). Then, in line 10, a fixed number of hypotheses are kept in the beam for the following stages of decoding. FEASIBLERERANKING() is a function that judges whether local re-ranking can be executed according to the following two constraints.

— The first constraint (lines 13–15): the maximum SBLM score of n-best hypotheses should be above a threshold s_{min}. This is because when the tree score is too low, it makes no sense to apply the syntax-based language model to re-rank the translation hypotheses.
— The second constraint (line 16): the length of source span for local re-ranking should be greater than l_{min}. It is motivated by the fact that the score of the subtree covering too few words does not always provide very reliable tree scoring results for re-ranking.

CKY-Style Decoding with Local Re-ranking

```
Input: a Chinese sentence $f$ and a syntax-based language model $P_{SBLM}(\cdot)$
Output: an English translation (with the underlying tree structure)
1 Function DECODINGWITHLR ($f$, $P_{SBLM}(\cdot)$)
2     for $l = 2$ to $m$ do                                          ◁ length of span
3       for $p = 1$ to $m - l + 1$ do                                ◁ begin of span
4         $q = p + l - 1$                                            ◁ end of span
5         for $k = p$ to $q - 1$ do                                  ◁ partition of span
6           $newhypotheses \leftarrow$ Compose($Q[p, k]$, $Q[k+1, q]$)
7           $Q[p, q] \leftarrow$ Update($Q[p, q]$, $newhypotheses$)
8         if FEASIBLERERANKING( $p, q, P_{SBLM}(\cdot)$) do
9           $Q[p, q] \leftarrow$ LocalReranking($Q[p, q]$, $P_{SBLM}(\cdot)$)
10       $Q[p, q] \leftarrow$ Top($Q[p,q]$, $beamsize$)              ◁ beam pruning
11    return Top($Q[p, q]$, 1)
12 Function FEASIBLERERANKING ( $p$, $q$, $P_{SBLM}(\cdot)$)
13    for each $t$ in $Q[p, q]$ do
14      if $P_{SBLM}(t) > maxscore$ do $maxscore \leftarrow P_{SBLM}(t)$
15    if $maxscore < s_{min}$ do return false                        ◁ constraint 1
16    if $q - p < l_{min}$ do return false                           ◁ constraint 2
17    return true
```

Figure 6 illustrates the local re-ranking algorithm. The dashed rounded box at the bottom of Figure 6 shows the case where the local re-ranking is activated, while the other dashed rounded box shows the case where the constraints of local re-ranking are not satisfied. As $l_{min} = 3$, the local re-ranking is not performed on the spans of length ≤ 3 (shaded chart cells in Figure 6).

5. ADAPTING SYNTAX-BASED LANGUAGE MODELS FOR MT

5.1 Syntax-Based Language Model Adaptation

In many machine learning (ML) and NLP tasks, it is natural to assume that the training and test data both follow the same distribution. However, for syntax-based language modeling in MT, the situation is not so pleasant. Typically, syntax-based language models are trained on the data of correct English (e.g., the Penn Treebank or sentences translated by human), while is tested on the output of MT system which is far from perfect English, even is incorrect English. This implies very different underlying distributions between training data (good English in Treebanks or bilingual

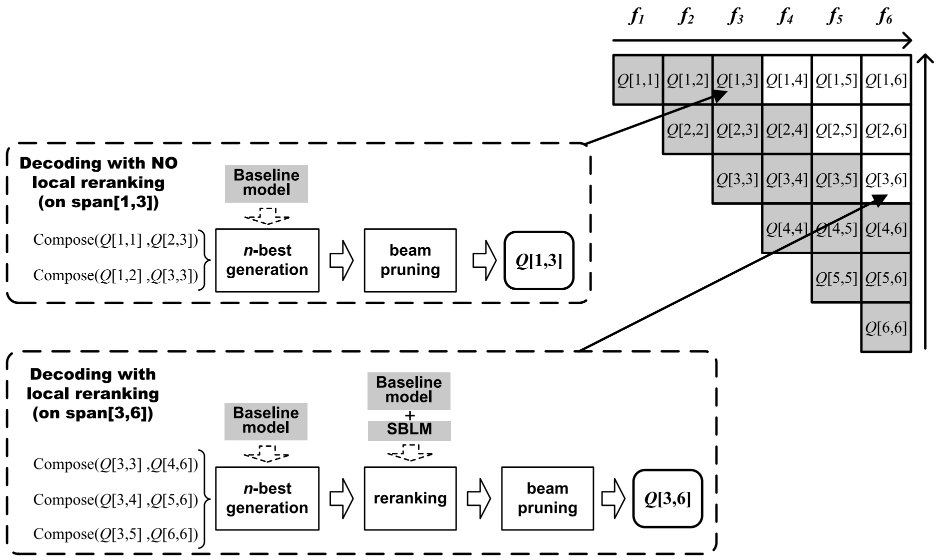


Fig. 6. Illustration of local re-ranking in CKY-style decoding (l_{min} = 3 and s_{min} = 0). The chart on the top right represents the structure that stores translation hypotheses in decoding. The shaded chart cells represent the spans on which the constraints of local re-ranking are not satisfied.

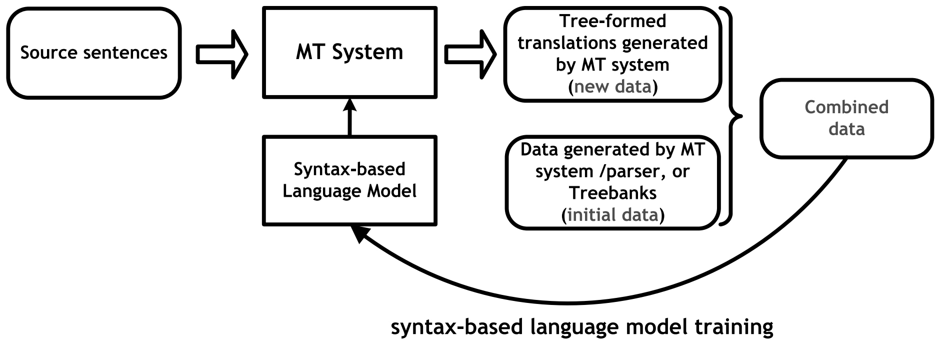

Source sentences
MT System
Tree-formed translations generated by MT system (new data)
Data generated by MT system /parser, or Treebanks (initial data)
Combined data
Syntax-based Language Model
syntax-based language model training

Fig. 7. Illustration of syntax-based language model adaptation for MT.

data) and test data (poor or even incorrect English in MT output). It is true even when their original texts are very close (e.g., in the same domain and writing-style). This problem probably results in unstable performance of syntax-based language models, especially when the MT output is far from correct English sentences.

The major reason behind this problem lies in that syntactic parsers are actually not designed for detecting the grammaticality in translations generated by MT systems [Koehn 2010]. To handle it, a straightforward solution is to manually annotate the MT outputs with syntactic structures and train syntax-based language models on the annotated MT translations. However, it is very expensive to create such a training data set. In most cases, it is even impractical to annotate enough data for training the language model. In this work we instead use a simpler and more practical method. The basic idea is similar to self-training for parser adaptation [McClosky et al. 2006], where a parser iteratively bootstraps itself using self-generated data. Differing from parser self-training, we bootstrap the syntax-based language model using the target-tree structures generated by MT. Figure 7 shows an intuitive illustration of our method, where initial data (or labeled data) refers to the standard training data used in syntax-language model training, such as Treebanks and the tree structures generated by the MT system or the syntactic parser, and new data (or unlabeled data) refers to the translations of a certain number of source sentences S.

In this method, we first train a base model using the initial data set L. We then use the syntax-based MT system to generate tree-formed translations $\{\pi_e\}$ for S. The newly-generated tree structures are treated as truth and combined with the initial data to train a new syntax-based language model. Though $\{\pi_e\}$ might not be well-formed English from the viewpoint of parsing, they are relatively more "preferred" by MT and thus more suitable for training syntax-based language models for translation tasks. If L is initialized with the MT-system-generated tree structures (for the target-side of the bilingual corpus), this procedure can be regarded as training on ill-formed MT output alone; otherwise, the resulting TSG-based LM is affected by the original treebank transformed TSG.

Like self-training, this procedure can be iterated, where the new model is used to generate new data in the next iteration. The below pseudocode summarizes this method, where T is the number of iterations, function TRANSLATE() generates the target-tree structures for a given set of source sentences, and function MODELTRAINING() returns a new syntax-based language model that is learned using the input training data. Note that, using this method, we could bootstrap any syntax-based language model no matter it is initially trained on the date generated using the MT system or on the treebank corpus.

Syntax-Based Language Model Adaptation for MT

Input: a syntax-based language model $P_{SBLM}(\cdot)$, a baseline syntax-based MT system MT, an initial data set L (tree structures generated by the MT system/parser for the target-side of the bilingual data or treebanks), a group of source sentence sets $\{S_i\}$ where each S_i consists of a certain number of source-language sentences

Output: a new syntax-based language model

```
1  Function ADAPTINGSBLM ($P_{SBLM}(\cdot)$, $MT$, $L$, $\{S_i\}$ )
2    $P_{SBLM}(\cdot)$ = MODELTRAINING ($L$)                 ◁ base model
3    $N = \varphi$
4    for $i = 1$ to $T$ do                                   ◁ iteration
5      $newdata$= TRANSLATE ($P_{SBLM}(\cdot)$, $MT$, $S_i$ )   ◁ new data for SBLM training
6      $N = N \cup newdata$                                  ◁ update $N$ with new data
7      $P_{SBLM}(\cdot)$ = MODELTRAINING ($L \cup N$)        ◁ train a new language model
8    return $P_{SBLM}(\cdot)$
9  Function MODELTRAINING ($L$)
10   return a new syntax-based language model trained on $L$
11 Function TRANSLATE ($P_{SBLM}(\cdot)$, $MT$, $S$ )
12   $\{\pi_e\}$ = generate tree-formed translations for $S$ using $MT$ and $P_{SBLM}(\cdot)$
13   return $\{\pi_e\}$
```

5.2 Controlling Overfitting

Previous studies have shown that big STSG rules are not very helpful for syntax-based machine translation [Cohn and Blunsom 2009; Liu and Gildea 2009]. In most cases, rare and big rules are considered to be noisy and generalize poorly on the unseen data (i.e., overfit the training data). This problem is even more severe when the (S)TSG is learned using the EM algorithm where the model is dominated by the biggest rules.

Although our method differs sequentially from the EM training with TSG[10], it would be very nice to have a mechanism that controls how likely the model overfits the training data.

A straightforward solution to this issue is to guide the model towards using frequent and small rules. This matches our intuition that good rules are small, with few internal nodes, frontier non-terminals and terminal strings [Cohn and Blunsom 2009]. To realize this idea, we use the variational Bayesian method [Beal 2003] as a replacement of the MLE in the training of TSG-based LM. Take PCFGs for instance. In MLE, the probability of a rule r is estimated according to the following equation:

$$\Pr(r|root(r)) = \frac{count(r)}{\sum_{r':root(r')=root(r)} count(r')}, \tag{8}$$

where $root(r)$ is a function that returns the root label of r. The variational Bayesian method slightly modifies the above equation and performs an inexact normalization, where the resulting rule probability will add up less than one, by passing counts through function $f(x) = \exp(\psi(x))$.

$$\Pr(r|root(r)) = \frac{f(count(r) + \alpha)}{f(\sum_{r':root(r')=root(r)} count(r') + \alpha)}, \tag{9}$$

where $\psi(x)$ is the digamma function [Johnson 2007]. It has the effect of subtracting 0.5 from its argument. The behavior of this estimation is determined by the choice of α. When α is set to a low value, it can be regarded as a way of "anti-smoothing". As about 0.5 is subtracted from the rule counts, small counts corresponding to rare rules (i.e., generally big rules) are penalized heavily, while large counts corresponding to frequent rules (i.e., generally small rules) are not be affected very much. As a result, low values of α make Equation (9) favor the rules which occur frequently and distrust the rules which occur rarely. In this way, the variational Bayesian method could control the overfitting caused by abusing rare and big rules. On the other hand, α can be set to a high value if we do not want to rule out any rules and smoothing is required. In this work, α is set to zero (the lowest value), since we wish to guide the model towards favoring frequent and small rules. In addition, after the normalization using variational Bayes, we perform an additional round of normalization without variational Bayes to add up rule probability to one.

6. EVALUATION

We carry out experiments on the NIST Chinese-English MT evaluation tasks. The system performance is evaluated in terms of the case-insensitive NIST version BLEU[%] (using the shortest reference length). Statistical significant test is conducted with the re-sampling method proposed by Koehn [2004].

[10]Both the EM algorithm and our proposed method (Section 5.1) follow the iterative framework to train TSGs. However, our method is substantially different from the general EM algorithm in several ways. In the EM training with TSGs, the objective is to maximize the likelihood of the data in terms of a probabilistic TSG [Knight 2009]. In contrast, our method (Figure 7) seeks the best tree structure (i.e., Viterbi output) using the MT system based on a joint model instead of the probabilistic TSG alone. Moreover, in our method, the MT system is optimized according to some metrics that measure the number of errors in MT output with respect to reference translations [Och 2003], which is different from the maximum-likelihood estimation used in EM. In addition, the MT system has sub-models (such as rule penalty) that eliminate the system bias towards the use of fewer but bigger rules [Marcu et al. 2006], while the EM training with TSG suffers from using too big rules and is prone to degenerate analyses of the data [Liu and Gildea 2009].

Table I. Data Sets Used

	# of words		# of sentences	
	Chinese	English	Chinese	English
Bilingual	3,166K	4,065K	138K	
Development set (4 refs)	4,821	21,637	336	336×4
Test set of MT04 (4 refs)	50,792	231,222	1,788	1,788×4
Test set of MT05 (4 refs)	31,030	138,239	1,082	1,082×4

6.1 Data Preparation

The bilingual training corpus consists of about 138K bilingual sentences (3.2M words on the source-language side and 4.1M words on the target-language side) extracted from the FBIS corpus[11]. To filter out noisy data, the bilingual sentences are extracted according to two criteria: first, we only select the sentence-pairs with the length ratio (length of target sentence/length of source sentence) ranging in (1/5, 5/1); and second, we only select the sentence-pairs with reasonable lexicon mapping probabilities[12] to prevent the extraction of incomplete sentences.

To obtain word-aligned corpus, GIZA++ is employed to perform bi-directional word alignment on the bilingual sentences. The "grow-diag-final-and" method [Koehn et al. 2003] is then used to generate the symmetric word alignments. The English side of the bilingual data is parsed using our re-implementation of Collins Model 2. The training data for our parser is Sections 02-21 of the *Wall Street Journal* (WSJ) Treebank. A 5-gram language model is trained on the English part of the LDC bilingual training data and the Xinhua portion of the Gigaword corpus. Our development set comes from the NIST MT 2003 evaluation corpus in which the sentences of more than 20 words are excluded in order to speed up the MERT. For test data, we choose the NIST MT evaluation corpora of 2004 and 2005. Table I shows the statistics of the data sets used in our experiments.

6.2 The MT System Setup

In our syntax-based MT system, both minimal GHKM rules [Galley et al. 2004] and SPMT rules [Marcu et al. 2006] are extracted from the bilingual corpus. Composed rules are generated by composing two or three minimal GHKM and SPMT rules. As described in Section 2.1, we use a CKY-style decoder with cube pruning [Huang and Chiang 2007] and beam search to translate new Chinese sentences. By default, the beam size is set to 30. We use MERT to optimize the weights of the log-linear model. Our MERT uses two stopping criteria: first, the weights do not change by more than 0.001; and second, the BLEU score does not change by more than 0.03%. Also, we stop the MERT run when the iteration number is more than 30 since we observe that the BLEU score is not really improved after the 30 first iterations.

To achieve state-of-the-art performance, we further advance our system in two ways: 1) We use four specialized translation modules to translate date, time, name, and by-line respectively, and then insert their translations into the SMT system.[13] 2) We

[11]LDC catalog number: LDC2003E14.

[12]To do this, we first identify the high-precision lexicon mapping using the Chinese-English Translation Lexicon Version 3.0 (LDC catalog number: LDC2002L27). We then calculate the probability by normalizing the count of high-precision lexicon mapping with the sentence length. Finally, we use the probabilities in both directions (source-to-target and target-to-source) to determine whether a sentence-pair can be chosen or not.

[13]These specialized translation modules are very effective in improving the BLEU scores of the SMT systems. For example, in our experiments, they lead to a +1.5~2.0 BLEU improvement for both our system and a phrase-based baseline *Moses*.

allow the use of explicit deletion rule[14] in decoding. Also, we design a feature (i.e., word-deletion feature) that gives a penalty exp(-1) to this rule, which allows the system to learn a preference for using more or less word deletion operations.

For language model integration, the size of *n*-best list (for the entire span) in traditional re-ranking is set to 1000. For local re-ranking, the size of *n*-best list (for all spans) is set to 30. Other parameters, such as s_{min} and l_{min} (Section 4.3), are optimized on the development set.

6.3 The Syntax-Based Language Model Setup

The treebank grammar-based language models and the TSG-based language models are both based on our in-house re-implementations of the PCFG model [Jurafsky and Martin 2008] and the Collins Model 2 [Collins 1999]. The Treebank Grammar-based language models are trained on the *WSJ* Treebank Section 01-21, while the TSG-based language models are trained on the target-side parse trees of the bilingual data. As our proposed model has a nice property that all grammar rules used in decoding are observed in the training of the model[15], it does not suffer from the "unseen" events and the bad (zero probability) estimates for the probability of grammar rules. Thus, for TSG-based language modeling, we train the PCFG model using MLE, and train the lexicalized model using the method described in Collins [1999] (with back-off estimates for only a few factors). By default, the syntax-based language model is incorporated as an integrated feature into MT decoding.

In addition, since our CKY-style MT decoder requires a binary synchronous grammar, a large number of virtual non-terminals are introduced by our rule binarizer. In some cases the decoder generates partial translations with incomplete subtrees rooting at a virtual non-terminal produced during binarization. This makes it difficult to score the subtree using the syntax-based language model because the non-terminal is never seen in language model training. To solve this problem, we modify the tree scoring function so that it can return a score for such type of subtree while ignoring the unknown root label as well as the corresponding incomplete tree structure.

6.4 Results

EXP1: The Proposed Language Model

First we evaluate the translations generated using various syntax-based language models, with a primary goal of studying the impact of syntax-based language model upon the translation task. Table II shows the results, where we also report the BLEU score of a state-of-the-art, open-source MT system *Moses* for comparison. As shown in Table II, stable BLEU improvements are achieved when TSG-based language models are incorporated into the baseline system. By using the lexicalized parsing model, the system generally obtains an improvement of more than 0.4 BLEU points. In contrast, the traditional Treebank grammar-based language models are not shown to be useful in improving BLEU scores, even leads to a -0.5 BLEU decline in some cases. These results confirm our motivation that the language modeling with TSGs is more appropriate for syntax-based MT than context-free Treebank grammars.

[14]The explicit deletion rule refers to the rule that directly translates a source word into NULL-word, such as **DT** → <的, NULL>.

[15]Note that, in addition to minimal GHKM rules, composed rules and SPMT rules are also used in our MT system. Theoretically, composed rules and SPMT rules are made up of minimal GHKM rules, and can be decomposed into sets of minimal rules [Galley et al. 2004, 2006; Marcu et al. 2006]. As minimal rules are defined as the basic units in the TSG-based language modeling, our model can use minimal rules to score any (target-language) tree structure generated by the MT system, no matter what types of rules are applied in generating the tree structure.

Table II. BLEU Scores of Various Syntax-Based Language Modeling Methods (*TG* and *TSG* Are Short for Treebank Grammar and Tree Substitution Grammar)

	BLEU4[%]		
	Dev	MT04	MT05
Moses	33.24	32.36	32.00
Baseline	35.15	34.67	34.03
+ TG-based LM (PCFG)	34.51 (-0.64)	34.21 (-0.46)	33.52 (-0.51)
+ TG-based LM (Lex)	34.65 (-0.50)	34.29 (-0.38)	33.51 (-0.52)
+ TSG-based LM (PCFG)	35.50 (+0.35)	34.89 (+0.22)	34.36 (+0.33)
+ TSG-based LM (Lex)	35.58 (+0.43)	35.06 (+0.39)	34.43 (+0.40)

Table III. BLEU Scores and Speeds of Various Language Model Integration Methods (Treebank Grammar-Based Language Modeling)

	BLEU4[%]			Speed (sentence/sec)
	Dev	MT04	MT05	
Baseline	35.15	34.67	34.03	0.102
+ Integrated feature	34.65 (-0.50)	34.29 (-0.38)	33.51 (-0.52)	0.056
+ Re-ranking (traditional)	34.87 (-0.28)	34.35 (-0.32)	33.94 (-0.09)	0.100
+ Local Re-ranking	34.89 (-0.26)	34.32 (-0.35)	33.83 (-0.20)	0.091

Table IV. BLEU Scores and Speeds of Various Language Model Integration Methods (Tree Substitution Grammar-Based Language Modeling)

	BLEU4[%]			Speed (sentence/sec)
	Dev	MT04	MT05	
Baseline	35.15	34.67	34.03	0.102
+ Integrated feature	35.58 (+0.43)	35.06 (+0.39)	34.43 (+0.40)	0.056
+ Re-ranking (traditional)	35.36 (+0.21)	34.86 (+0.19)	34.13 (+0.10)	0.097
+ Local Re-ranking	35.53 (+0.38)	35.15 (+0.48)	34.34 (+0.31)	0.084

EXP2: Syntax-Based Language Model Integration

We then investigate the impacts of different language model integration methods on BLEU score and translation speed. As the lexicalized parsing model is shown to be relatively more effective for language modeling than the PCFG model (Table II), we choose it for experimenting with language model integration in this set of experiments. Tables III and IV show the results. As expected, traditional re-ranking does not burden the system, while treating syntax-based LM as integrated feature degrades the translation speed greatly (about two times slower than the baseline)[16]. Also, traditional re-ranking seems to be not helpful in improving BLEU scores due to the redundancies in n-best lists. By contrast, local re-ranking finds a good "balance" between BLEU and speed. As shown in the tables, it is just 1.15∼1.20 times slower than the baseline. Comparing to employing SBLM as an integrated feature, it is 1.57∼1.60 times faster. More interestingly, as a "bonus" local re-ranking achieves comparable BLEU scores with its counterpart "integrated feature", even outperforms it in some cases. This result indicates that locally re-ranking translation hypotheses is a promising method for syntax-based language model integration. Thus we choose it as the default method for language model integration for the following experiments.

[16]The speed is measured on an Intel Core(TM) 2 Due E8500 3.16GHz CPU.

Table V. BLEU Scores of Language Models With and Without Adaptation
(The Training Set Is Initialized Using the Tree Structures Generated by the MT System)

	BLEU4[%] (Dev / MT04 / MT05)	
	No Adaptation	Adaptation
+ TSG-based LM (PCFG)	35.40 / 34.91 / 34.10	35.61 / 35.11 / 34.28
+ TSG-based LM (Lex)	35.46 / 35.10 / 34.35	35.73 / 35.21 / 34.49
Baseline	35.15 / 34.67 / 34.03	

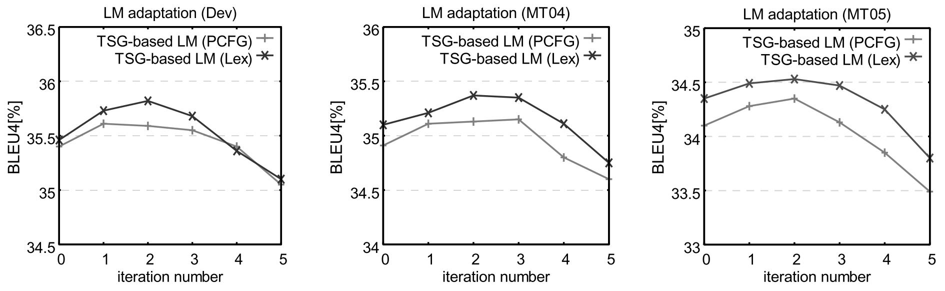


Fig. 8. BLEU curves of language model adaptation against iteration number (the training set is initialized using the tree structures generated by the MT system).

EXP3: Syntax-Based Language Model Adaptation

We also study the effect of our language model adaptation method on BLEU improvement. The additional (unlabeled) data used for language model adaptation is selected from a combined corpus of LDC2005T10, LDC2003E07 and LDC2005T06, which consists of approximately 220K sentence pairs from various news sources. To ease MT, some cleanups are performed. For example, we exclude the sentence pairs with length rate out of the range of (1/5, 5/1) and low lexicon mapping probabilities, as is described in Section 6.1. Also, we filter out hard samples on which our baseline system achieves a sentence-level BLEU score of lower than 35%[17]. We finally select 25K sentences from the source-side of the cleaned corpus. The corpus is then equally divided into five parts to perform five-iteration language model adaptation.

First, we investigate the effectiveness of the TSG-based language modeling on the ill-formed output. That is, we use the MT system to generate the tree structures for the target-side of the bilingual corpus, and train an initial TSG-based LM using these MT-system-generated tree structures. Then, the TSG-based LM is bootstrapped using the new data generated by the MT system, without combining the data with original Treebank transformed TSG grammar. Table V shows the difference in BLEU score when language model adaptation is performed for one iteration or not. We see a stable BLEU improvement when the syntax-based language models are trained on the mixture of initial data and newly-generated data. In most cases, the language model adaptation achieves an improvement of about 0.2 BLEU points. These results indicate that the syntax-based language models are able to benefit from the use of additional MT-preferred data.

We also attempt to find the optimal number of iterations for language model adaptation. Figure 8 shows that most improvement comes from the first two iterations. However, the improvement does not persist when more additional data are involved. As more iterations are performed, the improvement levels out quickly due to more

[17]This method can exclude the sentences with too many unknown words never seen in the training data.

Table VI. BLEU Scores of Language Models With and Without Adaptation (The Training Set Is Initialized Using the Tree Structures Generated by the Parser)

	BLEU4[%] (Dev / MT04 / MT05)	
	No Adaptation	Adaptation
+ TSG-based LM (PCFG)	35.57 / 35.01 / 34.30	35.82 / 35.19 / 34.41
+ TSG-based LM (Lex)	35.53 / 35.15 / 34.34	35.83 / 35.30 / 34.64
Baseline	35.15 / 34.67 / 34.03	

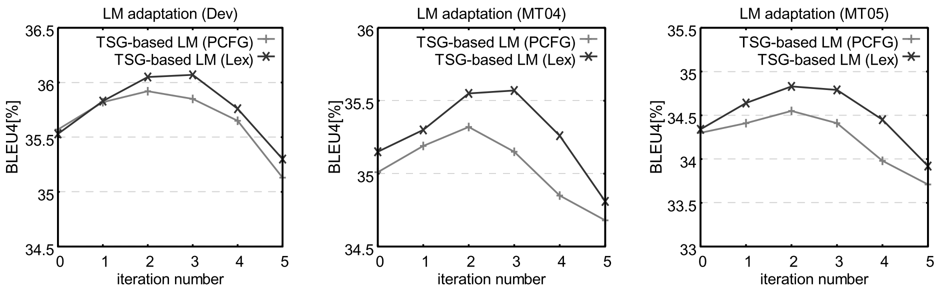


Fig. 9. BLEU curves of language model adaptation against iteration number (the training set is initialized using the tree structures generated by the parser).

bias and noises introduced. This observation is similar to that in [McClosky et al. 2006, 2008] where too much unlabeled data does not benefit parsing.

Then, we study the effectiveness of the TSG-based language modeling on the mixture of well-formed tree structures and ill-formed MT output. To this end, we initialize the training data for syntax-based LM using parse trees generated by the syntactic parser, as described in Section 3.3.1. We then bootstrap the TSG-based LM using the data combined with the new tree structures generated by the MT system. Table VI and Figure 9 show that language model adaptation is still useful in improving BLEU when mixed data is used. However, similar to the results shown in Figure 8, more iterations of bootstrapping does not help. Also as shown in Table VI, the best performance is achieved when the lexicalized TSG-based LM is used. Compared to the baseline, it obtains +0.6 BLEU improvements on all the evaluation sets, which are statistically significant at $p < 0.05$.

EXP4: Controlling Overfitting for TSG-Based Language Modeling

To examine the effect of controlling overfitting on language model adaptation, we apply the method described in Section 5.2 to the (un-lexicalized) TSG-based LM. Figures 10 and 11 show the results where the training set is initialized with the target-side tree structures generated by using the MT system (as in Figure 8) and the syntactic parser (as in Figure 9), respectively. As seen from the figures, the BLEU curves are not changed much when the variational Bayes is employed in the training of LM, which indicates that overfitting is not a very severe problem in our case. A possible reason for this phenomenon is that, in the method presented in Section 5.1, we use the MT system to generate the new tree structures for bootstrapping the TSG-based LM. Since the MT system has the capacity to eliminate the system bias towards derivations that use larger and fewer STSG rules, the training of TSG-based LM suffers little from the overfitting caused by abusing large TSG rules. In addition, as discussed in Section 5.2, another explanation might be that our TSG-based LM can generalize well on the unseen data since all target-sides of the STSG rules are estimated during the training of the TSG-based LM.

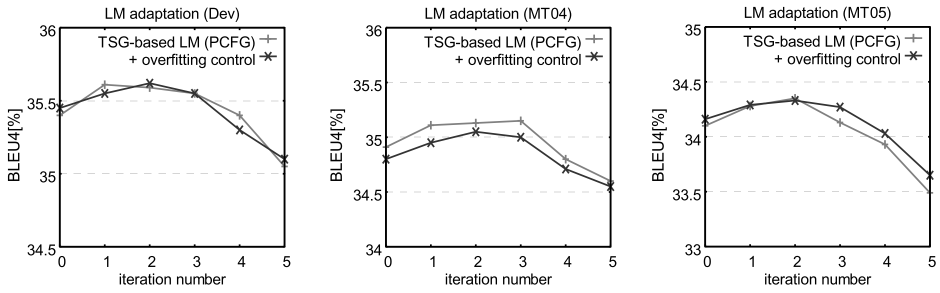


Fig. 10. BLEU curves of language model adaptation. The parameters are trained using the variational Bayesian method, and the training set is initialized using the tree structures generated by the MT system.

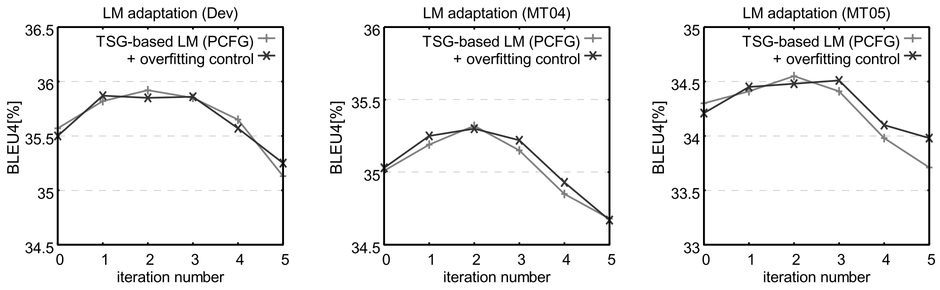


Fig. 11. BLEU curves of language model adaptation. The parameters are trained using the variational Bayesian method, and the training set is initialized using the tree structures generated by the parser.

Table VII. Comparison of Different TSGs for Language Modeling

	BLEU4[%]		
	Dev	MT04	MT05
LM using minimal rules only (CFG rules in the transformed tree structure)	35.57	35.01	34.30
+ composed rules and SPMT rules (TSG rules in the transformed tree structure)	35.77 (+0.20)	35.18 (+0.17)	34.26 (−0.04)

EXP5: Using More TSGs

We also study the effect of using more TSGs within language modeling. As described in Section 3.3.2, we enhance our approach by introducing rules that are consistent with the rule composing and SPMT extraction. That is, instead of training parameters from the CFG rules in the transformed tree structure, we estimate the rule probabilities using the TSG rules. As the Collins model cannot be trivially extended to handle TSGs, we only experiment with the non-lexicalized (PCFG) model for TSG-based language modeling here. Table VII shows that our system can benefit from more TSG rules used in language modeling. By the measure of BLEU, it obtains +0.2 BLEU improvements on the development set and the test set of MT04, while just degrades slightly on the test set of MT05.

EXP6: Recovering Internal Nodes

As discussed in Section 3.3.3, recovering internal nodes can lead to a more discriminative TSG for language modeling. To study its effect on our string-to-tree system, we compare the BLEU scores of the two TSG-based LMs with and without recovered internal nodes. Table VIII shows that the removal of internal nodes does not affect the

Table VIII. BLEU Scores of TSG-Based LMs With and Without Recovered Internal Nodes

	BLEU4[%]		
	Dev	MT04	MT05
Baseline (with internal nodes)	35.23	34.70	33.95
Baseline (without internal nodes)	35.15	34.67	34.03
+ TSG-based LM (PCFG) (with internal nodes)	35.28 (+0.05)	34.43 (-0.27)	33.72 (-0.23)
+ TSG-based LM (PCFG) (without internal nodes)	35.57 (+0.32)	35.01 (+0.34)	34.30 (+0.27)

Table IX. Data Sets Used in Large-Scale Experiments

	# of words		# of sentences	
	Chinese	English	Chinese	English
Bilingual	129M	147M	5.1M	
Development set (4 refs)	25,341	113,941	919	919 × 4
Test set of MT04 (4 refs)	50,792	231,222	1,788	1,788 × 4
Test set of MT05 (4 refs)	31,030	138,239	1,082	1,082 × 4

baseline MT system very much. However, it slightly degrades in BLEU score when the TSG-based LM (using non-lexicalized PCFG model) is integrated. This result indicates that recovering internal nodes does not really improve the TSG-based LM due to the data sparseness problem.

EXP7: Scaling to Large Data Sets

To test the effect of our TSG-based LM on larger data sets, we conduct experiments on another data configuration. This configuration uses the NIST portion of the bilingual training data available for the NIST 2008 track translation task[18]. In addition to the FBIS data used in the previous experiments, approximately five million new sentence-pairs are used. As a result, the size of the training corpus is increased by a factor of more than 30. To process the bilingual sentences of this large training corpus, we use the same methods as those described in Section 6.1. We also scale the development set to the full set of MT03 to obtain a stable convergence for MERT. Table IX shows the statistics of the date sets used in this set of experiments. We use all the bilingual data to obtain the STSG for our MT system. For syntax-based language modeling, we use all the target-side parse trees to train the PCFG model and use 358K sentences selected from LDC2003E14, LDC2005T10, LDC2003E07 and LDC2005T06 to train the lexicalized model. For language model adaptation, we randomly select 100K sentences from the data sets provided within CWMT08[19] as the unlabeled data.

Table X shows that the TSG-based LM is still effective when we switch to the large data set. Using all the techniques proposed in this article, the TSG-based LM obtains an improvement of about 0.5 BLEU scores on the development set and the test set of MT04, which is statistically significant at $p < 0.05$. Moreover, it is observed that the language model adaptation does not seem very helpful in improving the BLEU score. In most cases, its BLEU improvement is less than 0.2 points. The reason of this phenomenon might be due to the relatively small set of unlabeled data used here.

In addition, Table X shows the (normalized) weight of syntax-based language model (λ_{SBLM}) after MERT. For comparison, the (normalized) weight of *n*-gram language model (λ_{NLM}) is also reported. Compared to the TG-based LM, the TSG-based LM

[18]LDC catalog number: LDC2003E07, LDC2003E14, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85, and LDC2006E9.
[19]China Workshop on Machine Translation 2008. See http://nlpr-web.ia.ac.cn/cwmt-2008.

Table X. BLEU Scores of Various Syntax-Based Language Models on Large Data Sets

	BLEU4[%]			Weight	
	Dev	MT04	MT05	λ_{SBLM}	λ_{NLM}
Baseline	40.09	39.69	39.29	N/A	0.157
+ TG-based LM (PCFG)	39.41 (-0.68)	39.27 (-0.42)	38.94 (-0.35)	0.013	0.147
+ TG-based LM (Lex)	39.49 (-0.60)	39.21 (-0.48)	38.87 (-0.42)	0.011	0.150
+ TSG-based LM (PCFG)	40.50 (+0.41)	40.04 (+0.35)	39.27 (-0.02)	0.038	0.137
+ TSG-based LM (Lex)	40.54 (+0.45)	40.09 (+0.40)	39.43 (+0.14)	0.043	0.136
+ TSG-based LM (PCFG) + LM adaptation	40.51 (+0.42)	40.26 (+0.57)	39.52 (+0.23)	0.036	0.139
+ TSG-based LM (Lex) + LM adaptation	40.58 (+0.49)	40.34 (+0.65)	39.42 (+0.13)	0.045	0.137

Table XI. Results of Human Evaluation

	BLEU4 [%]	Judge 1	Judge 2	Judge 3	Judge Average
Baseline	35.15	2.90	3.12	3.18	3.07
Improved System	35.83	2.99	3.30	3.27	3.19
Human (reference)	100.00	4.56	4.67	4.88	4.70

gains much more importance in the syntax-based MT system, which further confirms the effectiveness of our approach.

6.5 Human Evaluation and Analysis

We also examine whether the improvements persist when the translation quality is judged by humans. To collect test data, we randomly select 50 sentences from the development set[20]. We develop a tiny program for interactive human evaluation. For each sentence, we first translate it using the baseline MT system and the improved system (TSG-based LM + language model adaptation), respectively. The translation results are then displayed as "MT output" on the screen through the program. We also display one of the reference translations as the third MT output, for the purpose of estimating the upper-bound performance. The three outputs are randomly ordered and displayed to the judges. Finally the other three reference translations are presented as the truth for reference.

Three judges participate in our experiment. They are all qualified translators who are skilled in reading and writing English. In the evaluation, they are required to carefully read the three MT translations and three reference translations, and assign a score from 1 to 5 to each MT output according its translation quality. Furthermore, the assessment of translation quality is required to focus on both grammatical fluency and translation accuracy of the outputs.

Table XI shows the evaluation results. We see that the improvement in human evaluation is consistent with the BLEU improvement. On average score, the improved system outperforms the baseline system more than 0.1 points, which is a statistically significant improvement at $p < 0.01$ ($t = 3.17$, $df = 149$). More interestingly, it is observed that the reference translation used in this study (provided by LDC) is not perfect as expected. It only achieves a score of 4.70. In some translations, obvious disfluency problems and even a few segments with ungrammatical or incorrect English are found. This observation is similar to that mentioned in Marcu et al. [2006].

[20]We choose the evaluation corpus of this size due to our limited human and finical resources.

We then analyze the data to study what problems and benefits the improved system has. We observe, first of all, that the overall grammaticality is not greatly improved at a glance. Several major problems with the baseline system still exist. For example, for the baseline system, the three most frequent error types are: incorrect noun (phrase) translation (20.1%), incorrect morphological generation (12.0%), incorrect word deletion (9.3%)[21]. They are still the most severe problems with the improved system, with nearly unchanged percentage numbers of error rate. We find that our proposed language model improves the MT output in several major aspects. For example, using our language model, the number of errors in verb sub-categorization is reduced from 13 to 6. Sample 1 shows an example where the baseline system chooses the wrong verb-categorization [**NP VP**] for the verb *require*, while the improved system chooses the correct verb sub-categorization [**NP** to **VP**]. Also, the structure movement is handled better. The number of this type of errors is reduced from 7 to 2. For example, in Sample 2, the improved system successfully handles the reordering pattern "提出 $\mathbf{NP}_1$ 的 $\mathbf{NNS}_2 \rightarrow$ propose $\mathbf{NNS}_2$ for $\mathbf{NP}_1$". In addition, verb form (e.g., plural or singular verb) is more appropriately generated. Over the test set, 13 out of 31 cases are improved. For example, in Samples 3 and 4, the improved system correctly chooses the plural forms for the verbs *have* and *be* according to their subjects which are plural nouns. These results indicate that our language model is more suitable for dealing with some specified problems, rather than handling every grammatical error in MT output.

Sample 1

Source:	***要求*** 上述 两 机构 以 重要性 为 序 ***列出*** 伊 尚未 完成 的 裁军 任务 。
Baseline:	It [*required*]$_{verb}$ [the above two agencies]$_{NP}$ [*list* the disarmament missions on the importance of the task to be completed by Iraq]$_{VP}$.
Improved:	It [*required*]$_{verb}$ [the above two agencies]$_{NP}$ [*to*]$_{TO}$ [*list* the disarmament missions on the importance of the task to be completed by Iraq]$_{VP}$.
Reference:	It [*required*]$_{verb}$ [the above-mentioned two agencies]$_{NP}$ [*to*]$_{TO}$ [*list* the disarmament missions yet to be completed by Iraq in sequence of importance]$_{VP}$.

Sample 2

Source:	我们 必须 ***提出*** 改组 美国 奥会 的 ***办法*** ...
Baseline:	We must [*set*]$_{verb}$ [the reorganization of the USOC]$_{NP}$ [*measures*]$_{NNS}$ ...
Improved:	We must [*propose*]$_{verb}$ [*methods*]$_{NNS}$ [for]$_{IN}$ [the reorganization of the USOC]$_{NP}$ ...
Reference:	We must [*propose*]$_{verb}$ [*ways*]$_{NNS}$ [to]$_{TO}$ [reorganize the USOC]$_{VP}$ ...

Sample 3

Source:	大约 二十 种 日本 贩售 的 游戏 ***具备*** 若干 线上 能力 。
Baseline:	[About 20]$_{QP}$ [games]$_{NNS}$ [sold in Japan]$_{VP}$ [*has*]$_{verb}$ several online capabilities.
Improved:	[About 20]$_{QP}$ [games]$_{NNS}$ [sold in Japan]$_{VP}$ [*have*]$_{verb}$ several online capabilities.
Reference:	[Nearly 20]$_{QP}$ [games]$_{NNS}$ [sold in Japan]$_{VP}$ [*have*]$_{verb}$ some sort of online capability.

Sample 4

Source:	但 比利时 小 镇 尼诺弗 汛情 依然 严重 ...
Baseline:	However, [floods]$_{NNS}$ [of a small town Ninove in Belgium]$_{PP}$ [*is*]$_{verb}$ still serious ...
Improved:	However, [floods]$_{NNS}$ [of a small town Ninove in Belgium]$_{PP}$ [*are*]$_{verb}$ still serious ...
Reference:	However, [floods]$_{NNS}$ [in small Belgian town Nover]$_{PP}$ [*are*]$_{verb}$ still serious ...

[21]The percentage number indicates the error rate which is approximately estimated by counting the appearance of errors in translations over the evaluation set.

7. RELATED WORK

In machine translation, researchers have been concerned for years about syntax-based language modeling. To our knowledge, the earliest attempt is Charniak et al. [2001] in which an English syntactic parser was employed to select the more grammatical translation output from a translation forest generated by a syntax-based MT system [Yamada and Knight 2001]. Charniak et al.'s [2001] method was basically a re-ranking method that re-sorted *n*-best list in terms of the model score produced by a parser. Though very simple and easy to implement, it suffers from the low quality of *n*-best list (or translation forest) due to the pruning at early stages of decoding. They showed that their method led to more grammatical outputs, but, unfortunately, degraded in BLEU score. As another representative work, Och et al. [2004] introduced the parse tree probability as a new feature into a (log-)linear-model-based re-ranking system for phrase-based MT. They reported that this feature is not helpful in improving BLEU scores of their phrase-based system. Cherry and Quirk [2008] investigated the question of whether a general syntactic parser is able to distinguish grammatical from ungrammatical texts. They trained a classifier using sentence length and model score produced by a parser with a Markovized, parent-annotated Context-Free Treebank Grammar, and then used the trained classifier to classify the *Wall Street Journal* sentences and self-made "negative" samples. Their results showed that such a syntax-based language model did not appear to be useful in classifying good and bad texts.

Although no promising BLEU improvements were achieved by using treebank grammar-based parsers as language models, there are studies that successfully integrated dependency parsing techniques into language modeling for machine translation, namely dependency language modeling. For example, Shen et al. [2008] extended Chiang [2005]'s hierarchical phrase-based approach with a dependency language model, and demonstrated a significant BLEU improvement over their strong baseline system. Another example is Post and Gildea [2008]. They advanced a Bracketing Transduction Grammar (BTG)-based system with the use of both the phrase structure parsing and dependency parsing techniques in language modeling. However, these studies all focused on improving the MT systems based on linguistically uninformed grammar[22]. In contrast, our work is on the basis of explicit syntax approaches in MT.

In the scenario of syntax-based MT, a number of studies have successfully exploited syntactic features for BLEU improvement. For example, several research groups have discussed the issue of synchronous tree scoring [Galley et al. 2006; Marcu et al. 2006; Mi and Huang 2008], where syntax-like-based features were introduced to model the generation of synchronous trees within their syntax-based systems. However, these features were trained using the synchronous grammar extracted from the bilingual data, and did not explicitly provide a mechanism to measure the degree of well-formedness of target-tree. More recently, Mi and Liu [2010] advanced their forest-based system using the dependency language model proposed in Shen et al. [2008], and demonstrated a promising BLEU improvement. On the other hand, Liu and Liu [2010] incorporated both the PCFG and lexicalized PCFG-based features into a tree-to-string system. They reported that the MT system could benefit from the joint learning of parsing and MT models. It is worth noting that, in a sense, Liu and Liu [2010] used a Treebank grammar-based parser to detect the well-formedness of the source-language parse trees yielding the input string. Differing from the results in previous work [Charniak et al. 2001; Och et al. 2004], they showed a great BLEU improvement over their baseline. This is because in Liu and Liu's work the parser's model score was

[22]This type of system is also called formally syntax-based system since it does not require the explicit syntax.

used to select better source parse trees for rule matching rather than directly searching for grammatically correct translations. While all these studies present promising results in introducing syntactic features into MT, it is still rare to see work that successfully advanced syntax-based MT systems using syntax-based language modeling beyond Treebank Grammar-based models.

Our work differs from previous work mainly in that we are concerned more with designing an effective language modeling method using tree substitution grammars for the improvement of syntax-based MT systems, and investigating methods of language model integration and adaptation for MT. Perhaps the most related work is Post and Gildea [2009]. They showed that language modeling with tree substitution grammars could achieve lower perplexities on the Penn Treebank than standard Context-Free grammars. Based on their experimental results, they further pointed out that this property was very feasible for applications like machine translation. However, Post and Gildea did not explicitly discuss the issue in the context of syntax-based MT. In addition to Post and Gildea's [2009] work, another line of research tries to improve MT with domain adaptation of *n*-gram language models [Bacchiani et al. 2004; Foster and Kuhn 2007; Koehn and Schroeder 2007; Snover et al. 2008; Zhao et al. 2004]. These studies showed that language model adaptation could significantly improve BLEU scores when MT systems performed on the out-of-domain data. As a matter of fact, like these works, we adapt the syntax-based language model from the Treebank data to the data used by MT. Beyond this, we bootstrap the syntax-based language model using the additional data generated by a syntax-based MT system. To our knowledge, the only previous work addressing this issue is Nakajima et al. [2002]. They adapted an *n*-gram language model with the data generated by a word-based MT system. By contrast, we focus on studying the issue in the context of syntax-based language modeling and experimenting with an MT system based on a state-of-the-art, string-to-tree model [Galley et al. 2004].

8. CONCLUSIONS

In this work we investigate syntax-based language modeling approaches for Chinese-English machine translation. We have presented a Tree Substitution Grammar-based language model to improve a state-of-the-art Chinese-English syntax-based MT system. By learning TSGs from the target-side of bilingual data, our model could better model the well-formedness of MT output than traditional Context-Free Treebank Grammar-based language models that are trained on the limited treebank data. On the NIST Chinese-English evaluation corpora, it achieves promising BLEU improvements over the baseline system. Moreover, we have presented three methods for efficient language model integration, as well as a simple and effective method for language model adaptation. Our experimental results show that these methods are very beneficial to the proposed language model, and consequently further speed-up the system and improve the translation accuracy. We expect that our promising results could encourage more studies on this interesting topic, such as exploring Tree Adjoining Grammars or other alternatives within language modeling.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their pertinent comments, Dr. Keh-Yih Su for the discussion on language model adaptation, Tongran Liu, Chunliang Zhang, and Shujie Yao for their valuable suggestions for improving the early version of this article, and Rushan Chen for developing parts of the baseline system.

REFERENCES

BACCHIANI, M., ROARK, B., AND SARACLAR, M. 2004. Language model adaptation with MAP estimation and the perceptron algorithm. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL'04)*. 21–24.

BEAL, M. 2003. Variational algorithms for approximate bayesian inference. Doctoral dissertation. University College London.

BONNEMA, R. 2002. *Probability models for DOP. Data-Oriented Parsing*. CSLI publications.

CHARNIAK, E. 1996. Tree-bank grammars. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*. 1031–1036.

CHARNIAK, E. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97)*. 598–603.

CHARNIAK, E. 2001. Immediate-head parsing for language models. In *Proceedings of the Association for Computational Linguistics (ACL'01)*. 124–131.

CHARNIAK, E., KNIGHT, K., AND YAMADA, K. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of the Machine Translation Summit IX (MT'03)*.

CHERRY, C. AND QUIRK, C. 2008. Discriminative, syntactic language modeling through latent SVMs. In *Proceedings of the Association for Machine Translation in the Americas (AMTA'08)*. 65–74.

CHIANG, D. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'05)*. 263–270.

CHIANG, D. AND KNIGHT, K. 2006. An introduction to synchronous grammars. In *Proceedings of the Association for Computational Linguistics (ACL'06)* (Tutorial).

COHN, T. AND BLUNSOM, P. 2009. A Bayesian model of syntax-directed tree to string grammar induction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'09)*. 352–361.

COLLINS, M. 1999. Head-driven statistical models for natural language parsing. PhD. dissertation. University of Pennsylvania.

DING, Y. AND PALMER, M. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the Association for Computational Linguistics (ACL'05)*. 541–548.

EISNER, J. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'03)*. 205–208.

FOSTER G. AND KUHN, R. 2007. Mixture-model adaptation for SMT. In *Proceedings of the 2nd Workshop on Statistical Machine Translation (SMT'07)*. 128–135.

GALLEY, M., HOPKINS, M., KNIGHT, K., AND MARCU, D. 2004. What's in a translation rule? In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04)*. 273–280.

GALLEY, M., GRAEHL, J., KNIGHT, K., MARCU, D., DENEEFE, S., WANG, W., AND THAYER, I. 2006. Scalable inferences and training of context-rich syntax translation models. In *Proceedings of the International Conference on Computer Linguistics (COLING'06)*. 961–968.

HUANG, L. AND CHIANG, D. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the Association for Computational Linguistics (ACL'07)*. 144–151.

JOHNSON, M. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'07)*. 296–305.

JURAFSKY, D. AND MARTIN, J. 2008. *Speech and Language Processing* 2nd Ed., Prentice-Hall, Upper Saddle River, NJ.

KNIGHT, K. 2009. *Bayesian Inference with Tears*. Tutorial Workbook.

KNIGHT, K. AND GRAEHL, J. 2005. An overview of probabilistic tree transducers for natural language processing. In *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistics (CICL'05)*. 1–24.

KNIGHT, K. AND KOEHN, P. 2009. *Topics in statistical machine translation*. In *Tutorial of the Association for Computational Linguistics (ACL'09)*.

KOEHN, P. 2004. Statistical Significance Tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*. 388–395.

KOEHN, P. 2010. *Statistical Machine Translation*. Cambridge University Press.

KOEHN, P. AND KNIGHT, K. 2003. Feature-rich statistical translation of noun phrases. In *Proceedings of the Association for Computational Linguistics (ACL'03)*. 311–318.

KOEHN, P. AND SCHROEDER, J. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the 2nd Workshop on Statistical Machine Translation (SMT'07)*. 224–227.

KOEHN, P., OCH, F., AND MARCU, D. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT-NAACL'03)*. 127–133.

LIU, D. AND GILDEA, D. 2009. Bayesian learning of phrasal tree-to-string templates. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'09)*. 1308–1317.

LIU, Y. AND LIU, Q. 2010. Joint parsing and translation. In *Proceedings of the International Conference on Computer Linguistics (COLING'10)*. 707–715.

LIU, Y., LIU, Q., AND LIN, S. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the International Conference on Computer Linguistics (COLING'06)*. 609–616.

MARCU, D., WANG, W., ECHIHABI, A., AND KNIGHT, K. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'06)*. 44–52.

MCCLOSKY, D., CHARNIAK, E., AND JOHNSON, M. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT-NAACL'06)*. 152–159.

MCCLOSKY, D., CHARNIAK, E., AND JOHNSON, M. 2008. When is self-training effective for parsing? In *Proceedings of the International Conference on Computer Linguistics (COLING'08)*. 561–568.

MI, H. AND HUANG, L. 2008. Forest-based translation rule extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*. 206–214.

MI, H. AND LIU, Q. 2010. Constituency to dependency translation with forests. In *Proceedings of the Association for Computational Linguistics (ACL'10)*. 1433–1442.

NAKAJIMA, H., YAMATOTO, H., AND WATANABE, T. 2002. Language model adaptation with additional text generated by machine translation. In *Proceedings of the International Conference on Computer Linguistics (COLING'02)*. 1–7.

OCH, F. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'03)*. 160–167.

OCH, F. AND NEY, H. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'02)*. 295–302.

OCH, F., GILDEA, D., KHUDANPUR, S., SARKAR, A., YAMADA, K., FRASER, A., KUMAR, S., SHEN, L., SMITH, D., ENG, K., JAIN, V., JIN, Z., AND RADEV, D. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04)*. 161–168.

POST, M. AND GILDEA, D. 2008. Parsers as language models for statistical machine translation. In *Proceedings of the 8th Conference of the Association for Machine Translation in the Americas (AMTA'08)*.

POST M. AND GILDEA, D. 2009. Language modeling with tree substitution grammars. In *Workshop on Grammar Induction, Representation of Language, and Language Learning (NIPS'09)*.

SHEN, L., XU, J., AND WEISCHEDEL, R. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the Association for Computational Linguistics (ACL'08)*. 577–585.

SNOVER, M., DORR, B., AND SCHWARTZ, R. 2008. Language and translation model adaptation using comparable corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*. 857–866.

YAMADA, K. AND KNIGHT, K. 2001. A syntax-based statistical machine translation model. In *Proceedings of the Association for Computational Linguistics (ACL'01)*. 132–139.

ZHAO, B., ECK, M., AND VOGEL, S. 2004. Language model adaptation for statistical machine translation via structured query models. In *Proceedings of the International Conference on Computer Linguistics (COLING'04)*. 411–417.

ZHANG, H., HUANG, L., GILDEA, D., AND KNIGHT, K. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT-NAACL'06)*. 256–263.

ZHANG, M., JIANG, H., AW, A., LI, H., TAN, C., AND LI, S. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of the Human Language Technology Conference (HLT'04)*. 559–567.

Received November 2010; revised February 2011; accepted April 2011