Soft Syntactic Constraints for Hierarchical Phrase-based Translation Using Latent Syntactic Distributions

Zhongqiang Huang
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
zqhuang@umiacs.umd.edu

Martin Čmejrek and **Bowen Zhou**
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{martin.cmejrek,zhou}@us.ibm.com

Abstract

In this paper, we present a novel approach to enhance hierarchical phrase-based machine translation systems with linguistically motivated syntactic features. Rather than directly using treebank categories as in previous studies, we learn a set of linguistically-guided latent syntactic categories automatically from a source-side parsed, word-aligned parallel corpus, based on the hierarchical structure among phrase pairs as well as the syntactic structure of the source side. In our model, each X nonterminal in a SCFG rule is decorated with a real-valued feature vector computed based on its distribution of latent syntactic categories. These feature vectors are utilized at decoding time to measure the similarity between the syntactic analysis of the source side and the syntax of the SCFG rules that are applied to derive translations. Our approach maintains the advantages of hierarchical phrase-based translation systems while at the same time naturally incorporates soft syntactic constraints.

1 Introduction

In recent years, syntax-based translation models (Chiang, 2007; Galley et al., 2004; Liu et al., 2006) have shown promising progress in improving translation quality, thanks to the incorporation of phrasal translation adopted from the widely used phrase-based models (Och and Ney, 2004) to handle local fluency and the engagement of synchronous context-free grammars (SCFG) to handle non-local phrase reordering. Approaches to syntax-based translation models can be largely categorized into two classes based on their dependency on annotated corpus (Chiang, 2007). Linguistically syntax-based models (e.g., (Yamada and Knight, 2001; Galley et al., 2004; Liu et al., 2006)) utilize structures defined over linguistic theory and annotations (e.g., Penn Treebank) and guide the derivation of SCFG rules with explicit parsing on at least one side of the parallel corpus. Formally syntax-based models (e.g., (Wu, 1997; Chiang, 2007)) extract synchronous grammars from parallel corpora based on the hierarchical structure of natural language pairs without any explicit linguistic knowledge or annotations. In this work, we focus on the hierarchical phrase-based models of Chiang (2007), which is formally syntax-based, and always refer the term SCFG, from now on, to the grammars of this model class.

On the one hand, hierarchical phrase-based models do not suffer from errors in syntactic constraints that are unavoidable in linguistically syntax-based models. Despite the complete lack of linguistic guidance, the performance of hierarchical phrase-based models is competitive when compared to linguistically syntax-based models. As shown in (Mi and Huang, 2008), hierarchical phrase-based models significantly outperform tree-to-string models (Liu et al., 2006; Huang et al., 2006), even when attempts are made to alleviate parsing errors using either forest-based decoding (Mi et al., 2008) or forest-based rule extraction (Mi and Huang, 2008).

On the other hand, when properly used, syntactic constraints can provide invaluable benefits to improve translation quality. The tree-to-string models of Mi and Huang (2008) can actually signif-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 138–147,
MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics

icantly outperform hierarchical phrase-based models when using forest-based rule extraction together with forest-based decoding. Chiang (2010) also obtained significant improvement over his hierarchical baseline by using syntactic parse trees on both source and target sides to induce fuzzy (not exact) tree-to-tree rules and by also allowing syntactically mismatched substitutions.

In this paper, we augment rules in hierarchical phrase-based translation systems with novel syntactic features. Unlike previous studies (e.g., (Zollmann and Venugopal, 2006)) that directly use explicit treebank categories such as NP, NP/PP (NP missing PP from the right) to annotate phrase pairs, we induce a set of latent categories to capture the syntactic dependencies inherent in the hierarchical structure of phrase pairs, and derive a real-valued feature vector for each X nonterminal of a SCFG rule based on the distribution of the latent categories. Moreover, we convert the equality test of two sequences of syntactic categories, which are either identical or different, into the computation of a similarity score between their corresponding feature vectors. In our model, two *symbolically different* sequences of syntactic categories could have a high similarity score in the feature vector representation if they are *syntactically similar*, and a low score otherwise. In decoding, these feature vectors are utilized to measure the similarity between the syntactic analysis of the source side and the syntax of the SCFG rules that are applied to derive translations. Our approach maintains the advantages of hierarchical phrase-based translation systems while at the same time naturally incorporates soft syntactic constraints. To the best of our knowledge, this is the first work that applies real-valued syntactic feature vectors to machine translation.

The rest of the paper is organized as follows. Section 2 briefly reviews hierarchical phrase-based translation models. Section 3 presents an overview of our approach, followed by Section 4 describing the hierarchical structure of aligned phrase pairs and Section 5 describing how to induce latent syntactic categories. Experimental results are reported in Section 6, followed by discussions in Section 7. Section 8 concludes this paper.

2 Hierarchical Phrase-Based Translation

An SCFG is a synchronous rewriting system generating source and target side string pairs simultaneously based on a context-free grammar. Each synchronous production (i.e., rule) rewrites a nonterminal into a pair of strings, γ and α, where γ (or α) contains terminal and nonterminal symbols from the source (or target) language and there is a one-to-one correspondence between the nonterminal symbols on both sides. In particular, the hierarchical model (Chiang, 2007) studied in this paper explores hierarchical structures of natural language and utilize only a unified nonterminal symbol X in the grammar,

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle$$

where $\sim$ is the one-to-one correspondence between X's in γ and α, and it can be indicated by underscripted co-indexes. Two example English-to-Chinese translation rules are represented as follows:

$$X \rightarrow \langle \text{give the pen to me}, \text{钢笔 给 我} \rangle \quad (1)$$

$$X \rightarrow \langle \text{give } X_1 \text{ to me}, X_1 \text{ 给 我} \rangle \quad (2)$$

The SCFG rules of hierarchical phrase-based models are extracted automatically from corpora of word-aligned parallel sentence pairs (Brown et al., 1993; Och and Ney, 2000). An aligned sentence pair is a tuple (E, F, A), where $E = e_1 \cdots e_n$ can be interpreted as an English sentence of length n, $F = f_1 \cdots f_m$ its translation of length m in a foreign language, and A a set of links between words of the two sentences. Figure 1 (a) shows an example of aligned English-to-Chinese sentence pair. Widely adopted in phrase-based models (Och and Ney, 2004), a pair of consecutive sequences of words from E and F is a *phrase pair* if all words are aligned only within the sequences and not to any word outside. We call a sequence of words a *phrase* if it corresponds to either side of a phrase pair, and a *non-phrase* otherwise. Note that the boundary words of a phrase pair may not be aligned to any other word. We call the phrase pairs with all boundary words aligned *tight* phrase pairs (Zhang et al., 2008). A tight phrase pair is the minimal phrase pair among all that share the same set of alignment links. Figure 1 (b) highlights the tight phrase pairs in the example sentence pair.

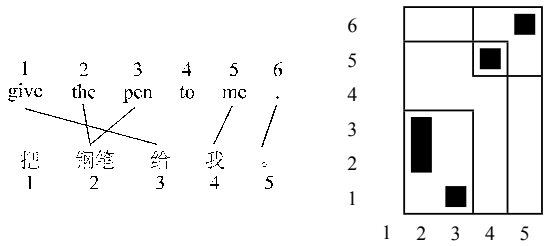

Figure 1: An example of word-aligned sentence pair (a) with tight phrase pairs marked in a matrix representation (b).

The extraction of SCFG rules proceeds as follows. In the first step, all phrase pairs below a maximum length are extracted as phrasal rules. In the second step, abstract rules are extracted from tight phrase pairs that contain other tight phrase pairs by replacing the sub phrase pairs with co-indexed X-nonterminals. Chiang (2007) also introduced several requirements (e.g., there are at most two nonterminals at the right hand side of a rule) to safeguard the quality of the abstract rules as well as keeping decoding efficient. In our example above, rule (2) can be extracted from rule (1) with the following sub phrase pair:

$$X \rightarrow \langle \text{the pen}, \text{钢笔} \rangle$$

The use of a unified X nonterminal makes hierarchical phrase-based models flexible at capturing non-local reordering of phrases. However, such flexibility also comes at the cost that it is not able to differentiate between different syntactic usages of phrases. Suppose rule $X \rightarrow \langle \text{I am reading } X_1, \cdots \rangle$ is extracted from a phrase pair with *I am reading a book* on the source side where X_1 is abstracted from the noun phrase pair . If this rule is used to translate *I am reading the brochure of a book fair*, it would be better to apply it over the entire string than over sub-strings such as *I ... the brochure of*. This is because the nonterminal X_1 in the rule was abstracted from a noun phrase on the source side of the training data and would thus be better (more informative) to be applied to phrases of the same type. Hierarchical phrase-based models are not able to distinguish syntactic differences like this.

Zollmann and Venugopal (2006) attempted to address this problem by annotating phrase pairs with treebank categories based on automatic parse trees. They introduced an extended set of categories (e.g., NP+V for *she went* and DT\NP for *great wall*, an noun phrase with a missing determiner on the left) to annotate phrase pairs that do not align with syntactic constituents. Their hard syntactic constraint requires that the nonterminals should match exactly to rewrite with a rule, which could rule out potentially correct derivations due to errors in the syntactic parses as well as to data sparsity. For example, NP cannot be instantiated with phrase pairs of type DT+NN, in spite of their syntactic similarity. Venugopal et al. (2009) addressed this problem by directly introducing soft syntactic preferences into SCFG rules using preference grammars, but they had to face the computational challenges of large preference vectors. Chiang (2010) also avoided hard constraints and took a soft alternative that directly models the cost of mismatched rule substitutions. This, however, would require a large number of parameters to be tuned on a generally small-sized held-out set, and it could thus suffer from over-tuning.

3 Approach Overview

In this work, we take a different approach to introduce linguistic syntax to hierarchical phrase-based translation systems and impose soft syntactic constraints between derivation rules and the syntactic parse of the sentence to be translated. For each phrase pair extracted from a sentence pair of a source-side parsed parallel corpus, we abstract its syntax by the sequence of highest root categories, which we call a *tag sequence*, that exactly[1] dominates the syntactic tree fragments of the source-side phrase. Figure 3 (b) shows the source-side parse tree of a sentence pair. The tag sequence for "the pen" is simply "NP" because it is a noun phrase, while phrase "give the pen" is dominated by a verb followed by a noun phrase, and thus its tag sequence is "VBP NP".

Let $TS = \{ts_1, \cdots, ts_m\}$ be the set of all tag sequences extracted from a parallel corpus. The syntax of each X nonterminal[2] in a SCFG rule can be then

[1] In case of a non-tight phrase pair, we only abstract and compare the syntax of the largest tight part.

[2] There are three X nonterminals (one on the left and two on the right) for binary abstract rules, two for unary abstract rules, and one for phrasal rules.

Tag Sequence	Probability
NP	0.40
DT NN	0.35
DT NN NN	0.25

Table 1: The distribution of tag sequences for X_1 in $X \rightarrow \langle$I am reading $X_1, \cdots \rangle$.

characterized by the distribution of tag sequences $\vec{P}_X(TS) = (p_X(ts_1), \cdots, p_X(ts_m))$, based on the phrase pairs it is abstracted from. Table 1 shows an example distribution of tag sequences for X_1 in $X \rightarrow \langle$I am reading $X_1, \cdots \rangle$.

Instead of directly using tag sequences, as we discussed their disadvantages above, we represent each of them by a real-valued feature vector. Suppose we have a collection of n latent syntactic categories $\mathcal{C} = \{c_1, \cdots, c_n\}$. For each tag sequence ts, we compute its distribution of latent syntactic categories $\vec{P}_{ts}(\mathcal{C}) = (p_{ts}(c_1), \cdots, p_{ts}(c_n))$. For example, $\vec{P}_{\text{"NP VP"}}(\mathcal{C}) = \{0.5, 0.2, 0.3\}$ means that the latent syntactic categories c_1, c_2, and c_3 are distributed as $p(c_1) = 0.5$, $p(c_2) = 0.2$, and $p(c_3) = 0.3$ for tag sequence "NP VP". We further convert the distribution to a normalized feature vector $\vec{F}(ts)$ to represent tag sequence ts:

$$\begin{aligned} \vec{F}(ts) &= (f_1(ts), \cdots, f_n(ts)) \\ &= \frac{(p_{ts}(c_1), \cdots, p_{ts}(c_n))}{\|(p_{ts}(c_1), \cdots, p_{ts}(c_n))\|} \end{aligned}$$

The advantage of using real-valued feature vectors is that the degree of similarity between two tag sequences ts and ts' in the space of the latent syntactic categories $\mathcal{C}$ can be simply computed as a dot-product[3] of their feature vectors:

$$\vec{F}(ts) \cdot \vec{F}(ts') = \sum_{1 \leq i \leq n} f_i(ts) f_i(ts')$$

which computes a syntactic similarity score in the range of 0 (totally syntactically different) to 1 (completely syntactically identical).

Similarly, we can represent the syntax of each X nonterminal in a rule with a feature vector $\vec{F}(X)$, computed as the sum of the feature vectors of tag sequences weighted by the distribution of tag sequences of the nonterminal X:

$$\vec{F}(X) = \sum_{ts \in TS} p_X(ts) \vec{F}(ts)$$

Now we can impose soft syntactic constraints using these feature vectors when a SCFG rule is used to translate a parsed source sentence. Given that a X nonterminal in the rule is applied to a span with tag sequence[4] ts as determined by a syntactic parser, we can compute the following syntax similarity feature:

$$\text{SynSim}(X, ts) = -\log(\vec{F}(ts) \cdot \vec{F}(X))$$

Except that it is computed on the fly, this feature can be used in the same way as the regular features in hierarchical translation systems to determine the best translation, and its feature weight can be tuned in the same way together with the other features on a held-out data set.

In our approach, the set of latent syntactic categories is automatically induced from a source-side parsed, word-aligned parallel corpus based on the hierarchical structure among phrase pairs along with the syntactic parse of the source side. In what follows, we will explain the two critical aspects of our approach, i.e., how to identify the hierarchical structures among all phrase pairs in a sentence pair, and how to induce the latent syntactic categories from the hierarchy to syntactically explain the phrase pairs.

4 Alignment-based Hierarchy

The aforementioned abstract rule extraction algorithm of Chiang (2007) is based on the property that a tight phrase pair can contain other tight phrase pairs. Given two non-disjoint tight phrase pairs that share at least one common alignment link, there are only two relationships: either one completely includes another or they do not include one another but have a non-empty overlap, which we call a non-trivial overlap. In the second case, the intersection, differences, and union of the two phrase pairs are

[3] Other measures such as KL-divergence in the probability space are also feasible.

[4] A normalized uniform feature vector is used for tag sequences (of parsed test sentences) that are not seen on the training corpus.

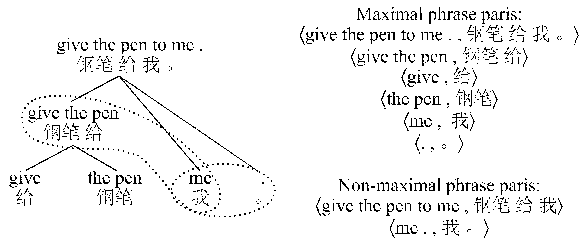

Figure 2: A decomposition tree of tight phrase pairs with all tight phrase pairs listed on the right. As highlighted, the two non-maximal phrase pairs are generated by consecutive sibling nodes.

also tight phrase pairs (see Figure 1 (b) for example), and the two phrase pairs, as well as their intersection and differences, are all sub phrase pairs of their union.

Zhang et al. (2008) exploited this property to construct a hierarchical decomposition tree (Bui-Xuan et al., 2005) of phrase pairs from a sentence pair to extract all phrase pairs in linear time. In this paper, we focus on learning the syntactic dependencies along the hierarchy of phrase pairs. Our hierarchy construction follows Heber and Stoye (2001).

Let $\mathcal{P}$ be the set of tight phrase pairs extracted from a sentence pair. We call a sequentially-ordered list[5] $L = (p_1, \cdots, p_k)$ of unique phrase pairs $p_i \in \mathcal{P}$ a *chain* if every two successive phrase pairs in L have a non-trivial overlap. A chain is *maximal* if it can not be extended to its left or right with other phrase pairs. Note that any sub-sequence of phrase pairs in a chain generates a tight phrase pair. In particular, chain L generates a tight phrase pair $\tau(L)$ that corresponds exactly to the union of the alignment links in $p \in L$. We call the phrase pairs generated by maximal chains *maximal* phrase pairs and call the other phrase pairs *non-maximal*. Non-maximal phrase pairs always overlap non-trivially with some other phrase pairs while maximal phrase pairs do not, and it can be shown that any non-maximal phrase pair can be generated by a sequence of maximal phrase pairs. Note that the largest tight phrase pair that includes all alignment links in A is also a maximal phrase pair.

[5]The phrase pairs can be sequentially ordered first by the boundary positions of the source-side phrase and then by the boundary positions of the target-side phrase.

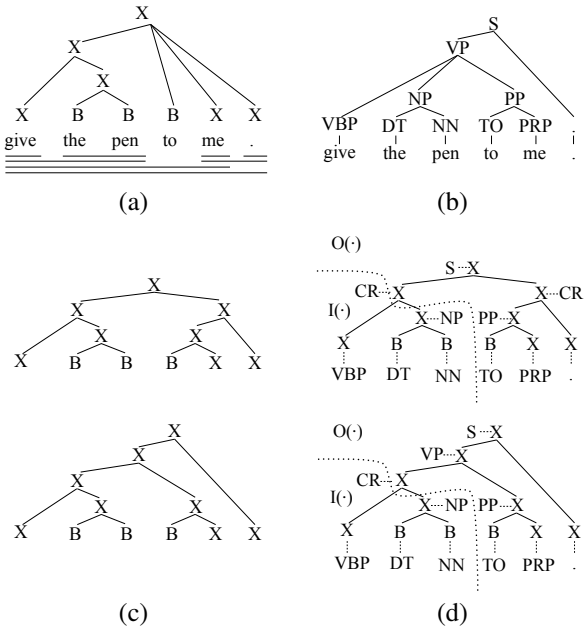

Figure 3: (a) decomposition tree for the English side of the example sentence pair with all phrases underlined, (b) automatic parse tree of the English side, (c) two example binarized decomposition trees with syntactic emissions in depicted in (d), where the two dotted curves give an example I(·) and O(·) that separate the forest into two parts.

Lemma 1 *Given two different maximal phrase pairs p_1 and p_2, exactly one of the following alternatives is true: p_1 and p_2 are disjoint, p_1 is a sub phrase pair of p_2, or p_2 is a sub phrase pair of p_1.*

A direct outcome of Lemma 1 is that there is an unique decomposition tree $T = (N, E)$ covering all of the tight phrase pairs of a sentence pair, where N is the set of maximal phrase pairs and E is the set of edges that connect between pairs of maximal phrase pairs if one is a sub phrase pair of another. All of the tight phrase pairs of a sentence pair can be extracted directly from the nodes of the decomposition tree (these phrase pairs are maximal), or generated by sequences of consecutive sibling nodes[6] (these phrase pairs are non-maximal). Figure 2 shows the decomposition tree as well as all of the tight phrase pairs that can be extracted from the example sentence pair in Figure 1.

We focus on the source side of the decomposition tree, and expand it to include all of the non-phrase

[6]Unaligned words may be added.

single words within the scope of the decomposition tree as frontiers and attach each as a child of the lowest node that contains the word. We then abstract the trees nodes with two symbol, X for phrases, and B for non-phrases, and call the result the *decomposition tree* of the source side phrases. Figure 3 (a) depicts such tree for the English side of our example sentence pair. We further recursively binarize[7] the decomposition tree into a binarized decomposition forest such that all phrases are directly represented as nodes in the forest. Figure 3 (c) shows two of the many binarized decomposition trees in the forest.

The binarized decomposition forest compactly encodes the hierarchical structure among phrases and non-phrases. However, the coarse abstraction of phrases with X and non-phrases with B provides little information on the constraints of the hierarchy. In order to bring in syntactic constraints, we annotate the nodes in the decomposition forest with syntactic observations based on the automatic syntactic parse tree of the source side. If a node aligns with a constituent in the parse tree, we add the syntactic category (e.g., NP) of the constituent as an emitted observation of the node, otherwise, it crosses constituent boundaries and we add a designated crossing category CR as its observation. We call the resulting forest a *syntactic decomposition forest*. Figure 3 (d) shows two syntactic decomposition trees of the forest based on the parse tree in Figure 3 (b). We will next describe how to learn finer-grained X and B categories based on the hierarchical syntactic constraints.

5 Inducing Latent Syntactic Categories

If we designate a unique symbol S as the new root of the syntactic decomposition forests introduced in the previous section, it can be shown that these forests can be generated by a probabilistic context-free grammar $G = (V, \Sigma, S, \mathcal{R}, \phi)$, where

- $V = \{S, X, B\}$ is the set of nonterminals,
- Σ is the set of terminals comprising treebank categories plus the CR tag (the crossing category),
- $S \in V$ is the unique start symbol,
- $\mathcal{R}$ is the union of the set of production rules each rewriting a nonterminal to a sequence of nonterminals and the set of emission rules each generating a terminal from a nonterminal,
- and ϕ assigns a probability score to each rule $r \in R$.

Such a grammar can be derived from the set of syntactic decomposition forests extracted from a source-side parsed parallel corpus, with rule probability scores estimated as the relative frequencies of the production and emission rules.

The X and B nonterminals in the grammar are coarse representations of phrase and non-phrases and do not carry any syntactic information at all. In order to introduce syntax to these nonterminals, we incrementally split[8] them into a set of latent categories $\{X_1, \cdots, X_n\}$ for X and another set $\{B_1, \cdots, B_n\}$ for B, and then learn a set of rule probabilities[9] ϕ on the latent categories so that the likelihood of the training forests are maximized. The motivation is to let the latent categories learn different preferences of (emitted) syntactic categories as well as structural dependencies along the hierarchy so that they can carry syntactic information. We call them *latent syntactic categories*. The learned X_i's represent syntactically-induced finer-grained categories of phrases and are used as the set of latent syntactic categories $\mathcal{C}$ described in Section 3. In related research, Matsuzaki et al. (2005) and Petrov et al. (2006) introduced latent variables to learn finer-grained distinctions of treebank categories for parsing, and Huang et al. (2009) used a similar approach to learn finer-grained part-of-speech tags for tagging. Our method is in spirit similar to these approaches.

Optimization of grammar parameters to maximize the likelihood of training forests can be achieved

[7] The intermediate binarization nodes are also labeled as either X or B based on whether they exactly cover a phrase or not.

[8] We incrementally split each nonterminal to 2, 4, 8, and finally 16 categories, with each splitting followed by several EM iterations to tune model parameters. We consider 16 an appropriate number for latent categories, not too small to differentiate between different syntactic usages and not too large for the extra computational and storage costs.

[9] Each binary production rule is now associated with a 3-dimensional matrix of probabilities, and each emission rule associated with a 1-dimensional array of probabilities.

by a variant of Expectation-Maximization (EM) algorithm. Recall that our decomposition forests are fully binarized (except the root). In the hypergraph representation (Huang and Chiang, 2005), the hyperedges of our forests all have the same format[10] $\langle (V, W), U \rangle$, meaning that node U expands to nodes V and W with production rule $U \rightarrow VW$. Given a forest F with root node R, we denote $e(U)$ the emitted syntactic category at node U and $\text{LR}(U)$ (or $\text{PL}(W)$, or $\text{PR}(V)$)[11] the set of node pairs (V, W) (or (U, V), or (U, W)) such that $\langle (V, W), U \rangle$ is a hyperedge of the forest. Now consider node U, which is either S, X, or B, in the forest. Let U_x be the latent syntactic category[12] of node U. We define $\text{I}(U_x)$ the part of the forest (includes $e(U)$ but not U_x) inside U, and $\text{O}(U_x)$ the other part of the forest (includes U_x but not $e(U)$) outside U, as illustrated in Figure 3 (d). The inside-outside probabilities are defined as:

$$\begin{aligned} \text{P}_{\text{IN}}(U_x) &= P(\text{I}(U_x)|U_x) \\ \text{P}_{\text{OUT}}(U_x) &= P(\text{O}(U_x)|S) \end{aligned}$$

which can be computed recursively as:

$$\begin{aligned} \text{P}_{\text{IN}}(U_x) &= \sum_{(V,W)\in \text{LR}(U)} \sum_{y,z} \begin{array}{c} \phi(U_x \rightarrow e(U)) \\ \times \phi(U_x \rightarrow V_y W_z) \\ \times \text{P}_{\text{IN}}(V_y)\text{P}_{\text{IN}}(W_z) \end{array} \\ \text{P}_{\text{OUT}}(U_x) &= \sum_{(V,W)\in \text{PL(U)}} \sum_{y,z} \begin{array}{c} \phi(V_y \rightarrow e(V)) \\ \times \phi(V_y \rightarrow W_z U_x) \\ \times \text{P}_{\text{OUT}}(V_y)\text{P}_{\text{IN}}(W_z) \end{array} \\ &+ \sum_{(V,W)\in \text{PR(U)}} \sum_{y,z} \begin{array}{c} \phi(V_y \rightarrow e(V)) \\ \times \phi(V_y \rightarrow U_x W_z) \\ \times \text{P}_{\text{OUT}}(V_y)\text{P}_{\text{IN}}(W_z) \end{array} \end{aligned}$$

In the E-step, the posterior probability of the occurrence of production rule[13] $U_x \rightarrow V_y W_z$ is computed as:

$$P(U_x \rightarrow V_y W_z|F) = \frac{\begin{array}{c} \phi(U_x \rightarrow e(U)) \\ \times \phi(U_x \rightarrow V_y W_z) \\ \times \text{P}_{\text{OUT}}(U_x)\text{P}_{\text{IN}}(V_y)\text{P}_{\text{IN}}(W_w) \end{array}}{\text{P}_{\text{IN}}(R)}$$

In the M-step, the expected counts of rule $U_x \rightarrow V_y W_z$ for all latent categories V_y and W_z are accumulated together and then normalized to obtain an update of the probability estimation:

$$\phi(U_x \rightarrow V_y W_z) = \frac{\#(U_x \rightarrow V_y W_z)}{\sum_{(V',W')} \sum_{y,z} \#(U_x \rightarrow V_y W_z)}$$

Recall that each node U labeled as X in a forest is associated with a phrase whose syntax is abstracted by a tag sequence. Once a grammar is learned, for each such node with a corresponding tag sequence ts in forest F, we compute the posterior probability that the latent category of node U being X_i as:

$$P(X_i|ts) = \frac{\text{P}_{\text{OUT}}(U_i)\text{P}_{\text{IN}}(U_i)}{\text{P}_{\text{IN}}(R)}$$

This contributes $P(X_i|ts)$ evidence that tag sequence ts belongs to a X_i category. When all of the evidences are computed and accumulated in $\#(X_i, ts)$, they can then be normalized to obtain the probability that the latent category of ts is X_i:

$$p_{ts}(X_i) = \frac{\#(X_i, ts)}{\sum_i \#(X_i, ts)}$$

As described in Section 3, the distributions of latent categories are used to compute the syntactic feature vectors for the SCFG rules.

6 Experiments

We conduct experiments on two tasks, English-to-German and English-to-Chinese, both aimed for speech-to-speech translation. The training data for the English-to-German task is a filtered subset of the Europarl corpus (Koehn, 2005), containing ∼300k parallel bitext with ∼4.5M tokens on each side. The dev and test sets both contain 1k sentences with one reference for each. The training data for the English-to-Chinese task is collected from transcription and human translation of conversations in travel domain. It consists of ∼500k parallel bitext with ∼3M tokens[14] on each side. Both dev and test sets contain ∼1.3k sentences, each with two references. Both

[10] The hyperedge corresponding to the root node has a different format because it is unary, but it can be handled similarly. When clear from context, we use the same variable to present both a node and its label.

[11] LR stands for the left and right children, PL for the parent and left children, and PR for the parent and right children.

[12] We never split the start symbol S, and denote $S_0 = S$.

[13] The emission rules can be handled similarly.

[14] The Chinese sentences are automatically segmented into words. However, BLEU scores are computed at character level for tuning and evaluation.

corpora are also preprocessed with punctuation removed and words down-cased to make them suitable for speech translation.

The baseline system is our implementation of the hierarchical phrase-based model of Chiang (2007), and it includes basic features such as rule and lexicalized rule translation probabilities, language model scores, rule counts, etc. We use 4-gram language models in both tasks, and conduct minimum-error-rate training (Och, 2003) to optimize feature weights on the dev set. Our baseline hierarchical model has 8.3M and 9.7M rules for the English-to-German and English-to-Chinese tasks, respectively.

The English side of the parallel data is parsed by our implementation of the Berkeley parser (Huang and Harper, 2009) trained on the combination of Broadcast News treebank from Ontonotes (Weischedel et al., 2008) and a speechified version of the WSJ treebank (Marcus et al., 1999) to achieve higher parsing accuracy (Huang et al., 2010). Our approach introduces a new syntactic feature and its feature weight is tuned in the same way together with the features in the baseline model. In this study, we induce 16 latent categories for both X and B nonterminals.

Our approach identifies $\sim$180k unique tag sequences for the English side of phrase pairs in both tasks. As shown by the examples in Table 2, the syntactic feature vector representation is able to identify similar and dissimilar tag sequences. For instance, it determines that the sequence of "DT JJ NN" is syntactically very similar to "DT ADJP NN" while very dissimilar to "NN CD VP". Notice that our latent categories are learned automatically to maximize the likelihood of the training forests extracted based on alignment and are not explicitly instructed to discriminate between syntactically different tag sequences. Our approach is not guaranteed to always assign similar feature vectors to syntactically similar tag sequences. However, as the experimental results show below, the latent categories are able to capture some similarities among tag sequences that are beneficial for translation.

Table 3 and 4 report the experimental results on the English-to-German and English-to-Chinese tasks, respectively. The addition of the syntax feature achieves a statistically significant improvement ($p \leq 0.01$) of 0.6 in BLEU on the test set of the English-to-German task. This improvement is substantial given that only one reference is used for each test sentence. On the English-to-Chinese task, the syntax feature achieves a smaller improvement of 0.41 BLEU on the test set. One potential explanation for the smaller improvement is that the sentences on the English-to-Chinese task are much shorter, with an average of only 6 words per sentence, compared to 15 words in the English-to-German task. The hypothesis space of translating a longer sentence is much larger than that of a shorter sentence. Therefore, there is more potential gain from using syntax features to rule out unlikely derivations of longer sentences, while phrasal rules might be adequate for shorter sentences, leaving less room for syntax to help as in the case of the English-to-Chinese task.

	Baseline	**+Syntax**	**Δ**
Dev	16.26	17.06	0.80
Test	16.41	17.01	0.60

Table 3: BLEU scores of the English-to-German task (one reference).

	Baseline	**+Syntax**	**Δ**
Dev	46.47	47.39	0.92
Test	45.45	45.86	0.41

Table 4: BLEU scores of the English-to-Chinese task (two references).

7 Discussions

The incorporation of the syntactic feature into the hierarchical phrase-based translation system also brings in additional memory load and computational cost. In the worst case, our approach requires storing one feature vector for each tag sequence and one feature vector for each nonterminal of a SCFG rule, with the latter taking the majority of the extra memory storage. We observed that about 90% of the X nonterminals in the rules only have one tag sequence, and thus the required memory space can be significantly reduced by only storing a pointer to the feature vector of the tag sequence for these nonterminals. Our approach also requires computing one dot-product of two feature vectors for each nonterminal when a SCFG rule is applied to a source span.

	Very similar $\vec{F}(ts) \cdot \vec{F}(ts') > 0.9$	**Not so similar** $0.4 \leq \vec{F}(ts) \cdot \vec{F}(ts') \leq 0.6$	**Very dissimilar** $\vec{F}(ts) \cdot \vec{F}(ts') < 0.1$
DT JJ NN	DT NN	DT JJ JJ NML NN	PP NP NN
	DT JJ JJ NN	DT JJ CC INTJ VB	NN CD VP
	DT ADJP NN	DT NN NN JJ	RB NP IN CD
VP	VB	VP PP JJ NN	JJ NN TO VP
	VB RB VB PP	VB NN NN VB	JJ WHNP DT NN
	VB DT DT NN	VB RB IN JJ	IN INTJ NP
ADJP	JJ	ADJP JJ JJ CC	ADJP IN NP JJ
	PDT JJ	ADJP VB JJ JJ	AUX RB ADJP
	RB JJ	ADVP WHNP JJ	ADJP VP

Table 2: Examples of similar and dissimilar tag sequences.

This cost can be reduced, however, by caching the dot-products of the tag sequences that are frequently accessed.

There are other successful investigations to impose soft syntactic constraints to hierarchical phrase-based models by either introducing syntax-based rule features such as the prior derivation model of Zhou et al. (2008) or by imposing constraints on translation spans at decoding time, e.g., (Marton and Resnik, 2008; Xiong et al., 2009; Xiong et al., 2010). These approaches are all orthogonal to ours and it is expected that they can be combined with our approach to achieve greater improvement.

This work is an initial effort to investigate latent syntactic categories to enhance hierarchical phrase-based translation models, and there are many directions to continue this line of research. First, while the current approach imposes soft syntactic constraints between the parse structure of the source sentence and the SCFG rules used to derive the translation, the real-valued syntactic feature vectors can also be used to impose soft constraints between SCFG rules when rule rewrite occurs. In this case, target side parse trees could also be used alone or together with the source side parse trees to induce the latent syntactic categories. Second, instead of using single parse trees during both training and decoding, our approach is likely to benefit from exploring parse forests as in (Mi and Huang, 2008). Third, in addition to the treebank categories obtained by syntactic parsing, lexical cues directly available in sentence pairs could also to explored to guide the learning of latent categories. Last but not the least, it would be interesting to investigate discriminative training approaches to learn latent categories that directly optimize on translation quality.

8 Conclusion

We have presented a novel approach to enhance hierarchical phrase-based machine translation systems with real-valued linguistically motivated feature vectors. Our approach maintains the advantages of hierarchical phrase-based translation systems while at the same time naturally incorporates soft syntactic constraints. Experimental results showed that this approach improves the baseline hierarchical phrase-based translation models on both English-to-German and English-to-Chinese tasks. We will continue this line of research and exploit better ways to learn syntax and apply syntactic constraints to machine translation.

Acknowledgements

This work was done when the first author was visiting IBM T. J. Watson Research Center as a research intern. We would like to thank Mary Harper for lots of insightful discussions and suggestions and the anonymous reviewers for the helpful comments.

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathemat-

ics of statistical machine translation: parameter estimation. *Computational Linguistics*.

Binh Minh Bui-Xuan, Michel Habib, and Christophe Paul. 2005. Revisiting T. Uno and M. Yagiura's algorithm. In *ISAAC*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*.

David Chiang. 2010. Learning to translate with source and target syntax. In *ACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2004. What's in a translation rule. In *HLT/NAACL*.

Steffen Heber and Jens Stoye. 2001. Finding all common intervals of k permutations. In *CPM*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *International Workshop on Parsing Technology*.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *EMNLP*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. A syntax-directed translator with extended domain of locality. In *CHSLP*.

Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. 2009. Improving a simple bigram hmm part-of-speech tagger by latent annotation and self-training. In *NAACL*.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable. In *EMNLP*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

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

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor, 1999. *Treebank-3*. Linguistic Data Consortium, Philadelphia.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *ACL*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *EMNLP*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *ACL*.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *ACL*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2009. Preference grammars: softening syntactic constraints to improve statistical machine translation. In *NAACL*.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, and Ann Houston, 2008. *OntoNotes Release 2.0*. Linguistic Data Consortium, Philadelphia.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*.

Deyi Xiong, Min Zhang, Aiti Aw, and Haizhou Li. 2009. A syntax-driven bracketing model for phrase-based translation. In *ACL-IJCNLP*.

Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Learning translation boundaries for phrase-based decoding. In *NAACL-HLT*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *ACL*.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting synchronous grammar rules from word-level alignments in linear time. In *COLING*.

Bowen Zhou, Bing Xiang, Xiaodan Zhu, and Yuqing Gao. 2008. Prior derivation models for formally syntax-based translation using linguistically syntactic parsing and tree kernels. In *SSST*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *StatMT*.