Inducing Word Alignments with Bilexical Synchronous Trees

Hao Zhang and **Daniel Gildea**
Computer Science Department
University of Rochester
Rochester, NY 14627

Abstract

This paper compares different bilexical tree-based models for bilingual alignment. EM training for the new model benefits from the dynamic programming "hook trick". The model produces improved dependency structure for both languages.

1 Introduction

A major difficulty in statistical machine translation is the trade-off between representational power and computational complexity. Real-world corpora for language pairs such as Chinese-English have complex reordering relationships that are not captured by current phrase-based MT systems, despite their state-of-the-art performance measured in competitive evaluations. Synchronous grammar formalisms that are capable of modeling such complex relationships while maintaining the context-free property in each language have been proposed for many years, (Aho and Ullman, 1972; Wu, 1997; Yamada and Knight, 2001; Melamed, 2003; Chiang, 2005), but have not been scaled to large corpora and long sentences until recently.

In Synchronous Context Free Grammars, there are two sources of complexity, grammar branching factor and lexicalization. In this paper we focus on the second issue, constraining the grammar to the binary-branching Inversion Transduction Grammar of Wu (1997). Lexicalization seems likely to help models predict alignment patterns between languages, and has been proposed by Melamed (2003) and implemented by Alshawi et al. (2000) and Zhang and Gildea (2005). However, each piece of lexical information considered by a model multiplies the number of states of dynamic programming algorithms for inference, meaning that we must choose how to lexicalize very carefully to control complexity.

In this paper we compare two approaches to lexicalization, both of which incorporate bilexical probabilities. One model uses bilexical probabilities across languages, while the other uses bilexical probabilities within one language. We compare results on word-level alignment, and investigate the implications of the choice of lexicalization on the specifics of our alignment algorithms. The new model, which bilexicalizes within languages, allows us to use the "hook trick" (Eisner and Satta, 1999) and therefore reduces complexity. We describe the application of the hook trick to estimation with Expectation Maximization (EM). Despite the theoretical benefits of the hook trick, it is not widely used in statistical monolingual parsers, because the savings do not exceed those obtained with simple pruning. We speculate that the advantages may be greater in an EM setting, where parameters to guide pruning are not (initially) available.

In order to better understand the model, we analyze its performance in terms of both agreement with human-annotated alignments, and agreement with the dependencies produced by monolingual parsers. We find that within-language bilexicalization does not improve alignment over cross-language bilexicalization, but does improve recovery of dependencies. We find that the hook trick significantly speeds training, even in the presence of pruning.

Section 2 describes the generative model. The hook trick for EM is explained in Section 3. In Section 4, we evaluate the model in terms of alignment error rate and dependency error rate. We conclude with discussions in Section 5.

Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 953–960,
Sydney, July 2006.

2 Bilexicalization of Inversion Transduction Grammar

The Inversion Transduction Grammar of Wu (1997) models word alignment between a translation pair of sentences by assuming a binary synchronous tree on top of both sides. Using EM training, ITG can induce good alignments through exploring the hidden synchronous trees from instances of string pairs.

ITG consists of unary production rules that generate English/foreign word pairs e/f:

$$X \rightarrow e/f$$

and binary production rules in two forms that generate subtree pairs, written:

$$X \rightarrow [Y\ Z]$$

and

$$X \rightarrow \langle Y\ Z \rangle$$

The square brackets indicate the right hand side rewriting order is the same for both languages. The pointed brackets indicate there exists a type of syntactic reordering such that the two right hand side constituents rewrite in the opposite order in the second language.

The unary rules account for the alignment links across two sides. Either e or f may be a special null word, handling insertions and deletions. The two kinds of binary rules (called straight rules and inverted rules) build up a coherent tree structure on top of the alignment links. From a modeling perspective, the synchronous tree that may involve inversions tells a generative story behind the word level alignment.

An example ITG tree for the sentence pair *Je les vois / I see them* is shown in Figure 1(left). The probability of the tree is the product rule probabilities at each node:

$$\begin{aligned} P(S \rightarrow A) & \\ & \cdot P(A \rightarrow [C\ B]) \\ & \cdot P(C \rightarrow I/Je) \\ & \cdot P(B \rightarrow \langle C\ C \rangle) \\ & \cdot P(C \rightarrow see/vois) \\ & \cdot P(C \rightarrow them/les) \end{aligned}$$

The structural constraint of ITG, which is that only binary permutations are allowed on each level, has been demonstrated to be reasonable by Zens and Ney (2003) and Zhang and Gildea (2004). However, in the space of ITG-constrained synchronous trees, we still have choices in making the probabilistic distribution over the trees more realistic. The original Stochastic ITG is the counterpart of Stochastic CFG in the bitext space. The probability of an ITG parse tree is simply a product of the probabilities of the applied rules. Thus, it only captures the fundamental features of word links and reflects how often inversions occur.

2.1 Cross-Language Bilexicalization

Zhang and Gildea (2005) described a model in which the nonterminals are lexicalized by English and foreign language word pairs so that the inversions are dependent on lexical information on the left hand side of synchronous rules. By introducing the mechanism of probabilistic head selection there are four forms of probabilistic binary rules in the model, which are the four possibilities created by taking the cross-product of two orientations (straight and inverted) and two head choices:

$$\begin{aligned} X(e/f) &\rightarrow [Y(e/f)\ Z] \\ X(e/f) &\rightarrow [Y\ Z(e/f)] \\ X(e/f) &\rightarrow \langle Y(e/f)\ Z \rangle \\ X(e/f) &\rightarrow \langle Y\ Z(e/f) \rangle \end{aligned}$$

where (e/f) is a translation pair.

A tree for our example sentence under this model is shown in Figure 1(center). The tree's probability is again the product of rule probabilities:

$$\begin{aligned} P(S \rightarrow A(see/vois)) & \\ & \cdot P(A(see/vois) \rightarrow [C\ B(see/vois)]) \\ & \cdot P(C \rightarrow C(I/Je)) \\ & \cdot P(B(see/vois) \rightarrow \langle C(see/vois)\ C \rangle) \\ & \cdot P(C \rightarrow C(them/les)) \end{aligned}$$

2.2 Head-Modifier Bilexicalization

One disadvantage of the model above is that it is not capable of modeling bilexical dependencies on the right hand side of the rules. Thus, while the probability of a production being straight or inverted depends on a bilingual word pair, it does not take head-modifier relations in either language into account. However, modeling complete bilingual bilexical dependencies as theorized in Melamed (2003) implies a huge parameter space of $O(|V|^2|T|^2)$, where $|V|$ and $|T|$ are the vocabulary sizes of the two languages. So, instead of modeling cross-language word translations and within-language word dependencies in

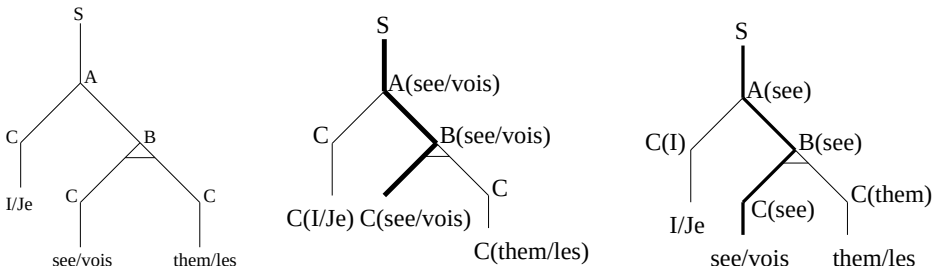

Figure 1: Parses for an example sentence pair under unlexicalized ITG (left), cross-language bilexicalization (center), and head-modifier bilexicaliztion (right). Thick lines indicate head child; crossbar indicates inverted production.

a joint fashion, we factor them apart. We lexicalize the dependencies in the synchronous tree using words from only one language and translate the words into their counterparts in the other language only at the bottom of the tree. Formally, we have the following patterns of binary dependency rules:

$$X(\mathbf{e}) \rightarrow [Y(\mathbf{e})\ Z(e')]$$

$$X(\mathbf{e}) \rightarrow [Y(e')\ Z(\mathbf{e})]$$

$$X(\mathbf{e}) \rightarrow \langle Y(\mathbf{e})\ Z(e')\rangle$$

$$X(\mathbf{e}) \rightarrow \langle Y(e')\ Z(\mathbf{e})\rangle$$

where $\mathbf{e}$ is an English head and e' is an English modifier.

Equally importantly, we have the unary lexical rules that generate foreign words:

$$X(e) \rightarrow e/f$$

To make the generative story complete, we also have a top rule that goes from the unlexicalized start symbol to the highest lexicalized nonterminal in the tree:

$$S \rightarrow X(e)$$

Figure 1(right), shows our example sentence's tree under the new model. The probability of a bilexical synchronous tree between the two sentences is:

$$\begin{aligned} P(S \rightarrow A(see)) \\ &\cdot P(A(\mathbf{see}) \rightarrow [C(I)\, B(\mathbf{see})]) \\ &\cdot P(C(I) \rightarrow I/Je) \\ &\cdot P(B(\mathbf{see}) \rightarrow \langle C(\mathbf{see})\, C(them)\rangle) \\ &\cdot P(C(see) \rightarrow see/vois) \\ &\cdot P(C(them) \rightarrow them/les) \end{aligned}$$

Interestingly, the lexicalized $B(\mathbf{see})$ predicts not only the existence of $C(them)$, but also that there is an inversion involved going from $C(\mathbf{see})$ to $C(them)$. This reflects the fact that direct object pronouns come after the verb in English, but before the verb in French. Thus, despite conditioning on information about words from only one language, the model captures syntactic reordering information about the specific language pair it is trained on. We are able to discriminate between the straight and inverted binary nodes in our example tree in a way that cross-language bilexicalization could not.

In terms of inferencing within the framework, we do the usual Viterbi inference to find the best bilexical synchronous tree and treat the dependencies and the alignment given by the Viterbi parse as the best ones, though mathematically the best alignment should have the highest probability marginalized over all dependencies constrained by the alignment. We do unsupervised training to obtain the parameters using EM. Both EM and Viterbi inference can be done using the dynamic programming framework of synchronous parsing.

3 Inside-Outside Parsing with the Hook Trick

ITG parsing algorithm is a CYK-style chart parsing algorithm extended to bitext. Instead of building up constituents over spans on a string, an ITG chart parser builds up constituents over subcells within a cell defined by two strings. We use $\beta(X(e), s, t, u, v)$ to denote the inside probability of $X(e)$ which is over the cell of (s, t, u, v) where (s, t) are indices into the source language string and (u, v) are indices into the target language string. We use $\alpha(X(e), s, t, u, v)$ to denote its outside probability. Figure 2 shows how smaller cells adjacent along diagonals can be combined to create a large cell. We number the subcells counterclockwise. To analyze the complexity of the algorithm with respect to input string

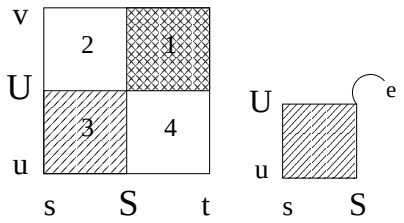

Figure 2: Left: Chart parsing over the bitext cell of (s, t, u, v). Right: One of the four hooks built for four corners for more efficient parsing.

length, without loss of generality, we ignore the nonterminal symbols X, Y, and Z to simplify the derivation.

The inside algorithm in the context of bilexical ITG is based on the following dynamic programming equation:

$$\begin{aligned}&\beta\,(e, s, t, u, v)\\ &= \sum_{S,U,e'} \begin{pmatrix} \beta_1(e) \cdot \beta_3(e') \cdot P([e'e] \mid e) \\ +\beta_2(e) \cdot \beta_4(e') \cdot P(\langle ee' \rangle \mid e) \\ +\beta_3(e) \cdot \beta_1(e') \cdot P([ee'] \mid e) \\ +\beta_4(e) \cdot \beta_2(e') \cdot P(\langle e'e \rangle \mid e) \end{pmatrix}\end{aligned}$$

So, on the right hand side, we sum up all possible ways (S, U) of splitting the left hand side cell and all possible head words (e') for the non-head subcell. e, e', s, t, u, v, S, and U all eight variables take $O(n)$ values given that the lengths of the source string and the target string are $O(n)$. Thus the entire DP algorithm takes $O(n^8)$ steps.

Fortunately, we can reduce the maximum number of interacting variables by factorizing the expression.

Let us keep the results of the summations over e' as:

$$\begin{aligned}\beta_1^+(e) &= \sum_{e'} \beta_1(e') \cdot P([ee'] \mid e)\\ \beta_2^+(e) &= \sum_{e'} \beta_2(e') \cdot P(\langle e'e \rangle \mid e)\\ \beta_3^+(e) &= \sum_{e'} \beta_3(e') \cdot P([e'e] \mid e)\\ \beta_4^+(e) &= \sum_{e'} \beta_4(e') \cdot P(\langle ee' \rangle \mid e)\end{aligned}$$

The computation of each β^+ involves four boundary indices and two head words. So, we can rely on DP to compute them in $O(n^6)$. Based on these intermediate results, we have the equivalent DP expression for computing inside probabilities:

$$\begin{aligned}&\beta\,(e, s, t, u, v)\\ &= \sum_{S,U} \begin{pmatrix} \beta_1(e) \cdot \beta_3^+(e) \\ +\beta_2(e) \cdot \beta_4^+(e) \\ +\beta_3(e) \cdot \beta_1^+(e) \\ +\beta_4(e) \cdot \beta_2^+(e) \end{pmatrix}\end{aligned}$$

We reduced one variable from the original expression. The maximum number of interacting variables throughout the algorithm is 7. So the improved inside algorithm has a time complexity of $O(n^7)$.

The trick of reducing interacting variables in DP for bilexical parsing has been pointed out by Eisner and Satta (1999). Melamed (2003) discussed the applicability of the so-called hook trick for parsing bilexical multitext grammars. The name hook is based on the observation that we combine the non-head constituent with the bilexical rule to create a special constituent that matches the head like a hook as demonstrated in Figure 2. However, for EM, it is not clear from their discussions how we can do the hook trick in the outside pass. The bilexical rules in all four directions are analogous. To simplify the derivation for the outside algorithm, we just focus on the first case: straight rule with right head word.

The outside probability of the constituent (e, S, t, U, v) in cell 1 being a head of such rules is:

$$\begin{aligned}&\sum_{s,u,e'} \left(\alpha(e) \cdot \beta_3(e') \cdot P([e'e] \mid e)\right)\\ &= \sum_{s,u} \left(\alpha(e) \cdot \left(\sum_{e'} \beta_3(e') \cdot P([e'e] \mid e)\right)\right)\\ &= \sum_{s,u} \Big(\alpha(e) \cdot \beta_3^+(e)\Big)\end{aligned}$$

which indicates we can reuse β^+ of the lower left neighbors of the head to make the computation feasible in $O(n^7)$.

On the other hand, the outside probability for (e', s, S, u, U) in cell 3 acting as a modifier of such

a rule is:

$$\sum_{t,v,e} \left(\alpha(e) \cdot \beta_1(e) \cdot P([e'e] \mid e)\right)$$
$$= \sum_{e} \left(P([e'e] \mid e) \cdot \left(\sum_{t,v} \alpha(e) \cdot \beta_1(e) \right) \right)$$
$$= \sum_{e} \Big(P([e', e] \mid e) \cdot \alpha_3^+(e) \Big)$$

in which we memorize another kind of intermediate sum to make the computation no more complex than $O(n^7)$.

We can think of α_3^+ as the outside probability of the hook on cell 3 which matches cell 1. Generally, we need outside probabilities for hooks in all four directions.

$$\alpha_1^+(e) = \sum_{s,u} \alpha(e) \cdot \beta_3(e)$$
$$\alpha_2^+(e) = \sum_{t,u} \alpha(e) \cdot \beta_4(e)$$
$$\alpha_3^+(e) = \sum_{t,v} \alpha(e) \cdot \beta_1(e)$$
$$\alpha_4^+(e) = \sum_{s,v} \alpha(e) \cdot \beta_2(e)$$

Based on them, we can add up the outside probabilities of a constituent acting as one of the two children of each applicable rule on top of it to get the total outside probability.

We finalize the derivation by simplifying the expression of the expected count of $(e \rightarrow [e'e])$.

$$EC(e \rightarrow [e'e])$$
$$= \sum_{s,t,u,v,S,U} \left(P([e'e] \mid e) \cdot \beta_3(e') \cdot \alpha(e) \cdot \beta_1(e)\right)$$
$$= \sum_{s,S,u,U} \left(P([e'e] \mid e) \cdot \beta_3(e') \cdot \left(\sum_{t,v} \alpha \cdot \beta_1 \right) \right)$$
$$= \sum_{s,S,u,U} \Big(P([e'e] \mid e) \cdot \beta_3(e') \cdot \alpha_3^+(e) \Big)$$

which can be computed in $O(n^6)$ as long as we have α_3^+ ready in a table. Overall we can do the inside-outside algorithm for the bilexical ITG in $O(n^7)$, by reducing a factor of n through intermediate DP.

The entire trick can be understood very clearly if we imagine the bilexical rules are unary rules that are applied on top of the non-head constituents to reduce it to a virtual lexical constituent (a hook) covering the same subcell while sharing the head word with the head constituent. However, if we build hooks looking for all words in a sentence whenever a complete constituent is added to the chart, we will build many hooks that are never used, considering that the words outside of larger cells are fewer and pruning might further reduce the possible outside words. Blind guessing of what might appear outside of the current cell will offset the saving we can achieve. Instead of actively building hooks, which are intermediate results, we can build them only when we need them and then cache them for future use. So the construction of the hooks will be invoked by the heads when the heads need to combine with adjacent cells.

3.1 Pruning and Smoothing

We apply one of the pruning techniques used in Zhang and Gildea (2005). The technique is general enough to be applicable to any parsing algorithm over bitext cells. It is called tic-tac-toe pruning since it involves an estimate of both the inside probability of the cell (how likely the words within the box in both dimensions are to align) and the outside probability (how likely the words outside the box in both dimensions are to align). By scoring the bitext cells and throwing away the bad cells that fall out of a beam, it can reduce over 70% of $O(n^4)$ cells using 10^{-5} as the beam ratio for sentences up to 25 words in the experiments, without harming alignment error rate, at least for the unlexicalized ITG.

The hook trick reduces the complexity of bilexical ITG from $O(n^8)$ to $O(n^7)$. With the tic-tac-toe pruning reducing the number of bitext cells to work with, also due to the reason that the grammar constant is very small for ITG. the parsing algorithm runs with an acceptable speed,

The probabilistic model has lots of parameters of word pairs. Namely, there are $O(|V|^2)$ dependency probabilities and $O(|V||T|)$ translation probabilities, where $|V|$ is the size of English vocabulary and $|T|$ is the size of the foreign language vocabulary. The translation probabilities of $P(f|X(e))$ are backed off to a uniform distribution. We let the bilexical dependency probabilities back off to uni-lexical dependencies in the following forms:

$$P([Y(*)\, Z(e')] \mid X(*))$$
$$P([Y(e')\, Z(*)] \mid X(*))$$
$$P(\langle Y(*)\, Z(e') \rangle \mid X(*))$$
$$P(\langle Y(e')\, Z(*) \rangle \mid X(*))$$

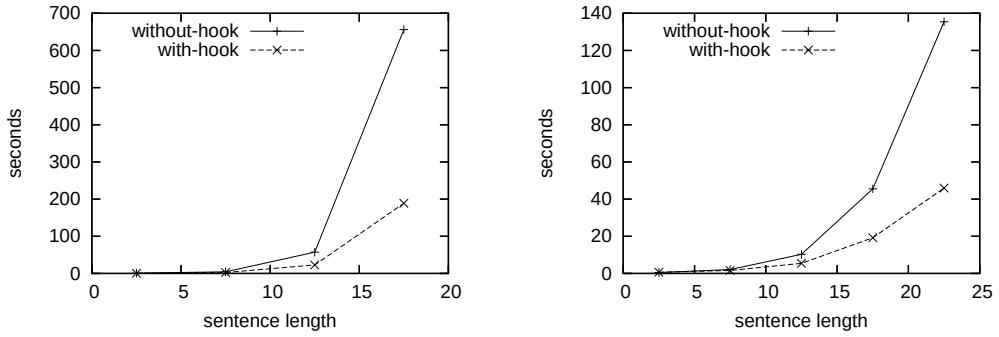

Figure 3: Speedup for EM by the Hook Trick. (a) is without pruning. In (b), we apply pruning on the bitext cells before parsing begins.

The two levels of distributions are interpolated using a technique inspired by Witten-Bell smoothing (Chen and Goodman, 1996). We use the expected count of the left hand side lexical nonterminal to adjust the weight for the EM-trained bilexical probability. For example,

$$P([Y(e)\ Z(e')] \mid X(e)) = (1-\lambda)P_{EM}([Y(e)\ Z(e')] \mid X(e)) + \lambda P([Y(*)\ Z(e')] \mid X(*))$$

where

$$\lambda = 1/(1 + Expected\_Counts(X(e)))$$

4 Experiments

First of all, we are interested in finding out how much speedup can be achieved by doing the hook trick for EM. We implemented both versions in C++ and turned off pruning for both. We ran the two inside-outside parsing algorithms on a small test set of 46 sentence pairs that are no longer than 25 words in both languages. Then we put the results into buckets of $(1-4)$, $(5-9)$, $(10-14)$, $(15-19)$, and $(20-24)$ according to the maximum length of two sentences in each pair and took averages of these timing results. Figure 3 (a) shows clearly that as the sentences get longer the hook trick is helping more and more. We also tried to turn on pruning for both, which is the normal condition for the parsers. Both are much faster due to the effectiveness of pruning. The speedup ratio is lower because the hooks will less often be used again since many cells are pruned away. Figure 3 (b) shows the speedup curve in this situation.

We trained both the unlexicalized and the lexicalized ITGs on a parallel corpus of Chinese-English newswire text. The Chinese data were automatically segmented into tokens, and English capitalization was retained. We replaced words occurring only once with an unknown word token, resulting in a Chinese vocabulary of 23,783 words and an English vocabulary of 27,075 words.

We did two types of comparisons. In the first comparison, we measured the performance of five word aligners, including IBM models, ITG, the lexical ITG (LITG) of Zhang and Gildea (2005), and our bilexical ITG (BLITG), on a hand-aligned bilingual corpus. All the models were trained using the same amount of data. We ran the experiments on sentences up to 25 words long in both languages. The resulting training corpus had 18,773 sentence pairs with a total of 276,113 Chinese words and 315,415 English words.

For scoring the Viterbi alignments of each system against gold-standard annotated alignments, we use the alignment error rate (AER) of Och and Ney (2000), which measures agreement at the level of pairs of words:

$$AER = 1 - \frac{|A \cap G_P| + |A \cap G_S|}{|A| + |G_S|}$$

where A is the set of word pairs aligned by the automatic system, G_S is the set marked in the gold standard as "sure", and G_P is the set marked as "possible" (including the "sure" pairs). In our Chinese-English data, only one type of alignment was marked, meaning that $G_P = G_S$.

In our hand-aligned data, 47 sentence pairs are no longer than 25 words in either language and were used to evaluate the aligners.

A separate development set of hand-aligned sentence pairs was used to control overfitting. The subset of up to 25 words in both languages was used. We chose the number of iterations for EM

	Precision	*Recall*	*Alignment Error Rate*
IBM-1	.56	.42	.52
IBM-4	.67	.43	.47
ITG	.68	.52	.41
LITG	.69	.51	.41
BLITG	.68	.51	.42

	Precision	*Recall*	*Dependency Error Rate*
ITG-lh	.11	.11	.89
ITG-rh	.22	.22	.78
LITG	.13	.12	.88
BLITG	.24	.22	.77

Table 1: Bilingual alignment and English dependency results on Chinese-English corpus ($\leq$ 25 words on both sides). LITG stands for the cross-language Lexicalized ITG. BLITG is the within-English Bilexical ITG. ITG-lh is ITG with left-head assumption on English. ITG-rh is with right-head assumption.

	Precision	*Recall*	*AER*
ITG	.59	.60	.41
LITG	.60	.57	.41
BLITG	.58	.55	.44

	Precision	*Recall*	*DER*
ITG-rh	.23	.23	.77
LITG	.11	.11	.89
BLITG	.24	.24	.76

Table 2: Alignment and dependency results on a larger Chinese-English corpus.

training as the turning point of AER on the development data set. The unlexicalized ITG was trained for 3 iterations. LITG was trained for only 1 iteration, partly because it was initialized with fully trained ITG parameters. BLITG was trained for 3 iterations.

For comparison, we also included the results from IBM Model 1 and Model 4. The numbers of iterations for the training of the IBM models were also chosen to be the turning points of AER changing on the development data set.

We also want to know whether or not BLITG can model dependencies better than LITG. For this purpose, we also used the AER measurement, since the goal is still getting higher precision/recall for a set of recovered word links, although the dependency word links are within one language. For this reason, we rename AER to Dependency Error Rate. Table 1(right) is the dependency results on English side of the test data set. The dependency results on Chinese are similar.

The gold standard dependencies were extracted from Collins' parser output on the sentences. The LITG and BLITG dependencies were extracted from the Viterbi synchronous trees by following the head words.

For comparison, we also included two base-line results. ITG-lh is unlexicalized ITG with left-head assumption, meaning the head words always come from the left branches. ITG-rh is ITG with right-head assumption.

To make more confident conclusions, we also did tests on a larger hand-aligned data set used in Liu et al. (2005). We used 165 sentence pairs that are up to 25 words in length on both sides.

5 Discussion

The BLITG model has two components, namely the dependency model on the upper levels of the tree structure and the word-level translation model at the bottom. We hope that the two components will mutually improve one another. The current experiments indicate clearly that the word level alignment does help inducing dependency structures on both sides. The precision and recall on the dependency retrieval sub-task are almost doubled for both languages from LITG which only has a kind of uni-lexical dependency in each language. Although 20% is a low number, given the fact that the dependencies are learned basically through contrasting sentences in two languages, the result is encouraging. The results slightly improve over ITG with right-head assumption for English, which is based on linguistic insight. Our results also echo the findings of Kuhn (2004). They found that based on the guidance of word alignment between English and multiple other languages, a modified EM training for PCFG on English can bootstrap a more accurate monolingual probabilistic parser. Figure 4 is an example of the dependency tree on the English side from the output of BLITG, comparing against the parser output.

We did not find that the feedback from the de-

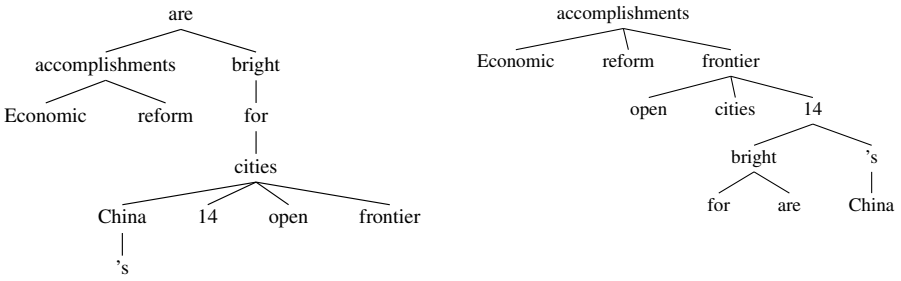

Figure 4: Dependency tree extracted from parser output vs. Viterbi dependency tree from BLITG

pendencies help alignment. To get the reasons, we need further and deeper analysis. One might guess that the dependencies are modeled but are not yet strong and good enough given the amount of training data. Since the training algorithm EM has the problem of local maxima, we might also need to adjust the training algorithm to obtain good parameters for the alignment task. Initializing the model with good dependency parameters is a possible adjustment. We would also like to point out that alignment task is simpler than decoding where a stronger component of reordering is required to produce a fluent English sentence. Investigating the impact of bilexical dependencies on decoding is our future work.

Acknowledgments This work was supported by NSF ITR IIS-09325646 and NSF ITR IIS-0428020.

References

Albert V. Aho and Jeffery D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1):45–60.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Conference of the Association for Computational Linguistics (ACL-96)*, pages 310–318, Santa Cruz, CA. ACL.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, pages 263–270, Ann Arbor, Michigan.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *37th Annual Meeting of the Association for Computational Linguistics*.

Jonas Kuhn. 2004. Experiments in parallel-text based grammar induction. In *Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics (ACL-04)*.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, Ann Arbor, Michigan.

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, Edmonton.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Conference of the Association for Computational Linguistics (ACL-00)*, pages 440–447, Hong Kong, October.

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

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Conference of the Association for Computational Linguistics (ACL-01)*, Toulouse, France.

Richard Zens and Hermann Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.

Hao Zhang and Daniel Gildea. 2004. Syntax-based alignment: Supervised or unsupervised? In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, Geneva, Switzerland, August.

Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, Ann Arbor, MI.