Bayesian Learning of Non-compositional Phrases with Synchronous Parsing

Hao Zhang
Computer Science Department
University of Rochester
Rochester, NY 14627
zhanghao@cs.rochester.edu

Chris Quirk
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
chrisq@microsoft.com

Robert C. Moore
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
bobmoore@microsoft.com

Daniel Gildea
Computer Science Department
University of Rochester
Rochester, NY 14627
gildea@cs.rochester.edu

Abstract

We combine the strengths of Bayesian modeling and synchronous grammar in unsupervised learning of basic translation phrase pairs. The structured space of a synchronous grammar is a natural fit for phrase pair probability estimation, though the search space can be prohibitively large. Therefore we explore efficient algorithms for pruning this space that lead to empirically effective results. Incorporating a sparse prior using Variational Bayes, biases the models toward generalizable, parsimonious parameter sets, leading to significant improvements in word alignment. This preference for sparse solutions together with effective pruning methods forms a phrase alignment regimen that produces better end-to-end translations than standard word alignment approaches.

1 Introduction

Most state-of-the-art statistical machine translation systems are based on large phrase tables extracted from parallel text using word-level alignments. These word-level alignments are most often obtained using Expectation Maximization on the conditional generative models of Brown et al. (1993) and Vogel et al. (1996). As these word-level alignment models restrict the word alignment complexity by requiring each target word to align to zero or one source words, results are improved by aligning both source-to-target as well as target-to-source, then heuristically combining these alignments. Finally, the set of phrases consistent with the word alignments are extracted from every sentence pair; these form the basis of the decoding process. While this approach has been very successful, poor word-level alignments are nonetheless a common source of error in machine translation systems.

A natural solution to several of these issues is unite the word-level and phrase-level models into one learning procedure. Ideally, such a procedure would remedy the deficiencies of word-level alignment models, including the strong restrictions on the form of the alignment, and the strong independence assumption between words. Furthermore it would obviate the need for heuristic combination of word alignments. A unified procedure may also improve the identification of non-compositional phrasal translations, and the attachment decisions for unaligned words.

In this direction, Expectation Maximization at the phrase level was proposed by Marcu and Wong (2002), who, however, experienced two major difficulties: computational complexity and controlling overfitting. Computational complexity arises from the exponentially large number of decompositions of a sentence pair into phrase pairs; overfitting is a problem because as EM attempts to maximize the likelihood of its training data, it prefers to directly explain a sentence pair with a single phrase pair.

In this paper, we attempt to address these two issues in order to apply EM above the word level.

Proceedings of ACL-08: HLT, pages 97–105,
Columbus, Ohio, USA, June 2008. ©2008 Association for Computational Linguistics

We attack computational complexity by adopting the polynomial-time Inversion Transduction Grammar framework, and by only learning small *non-compositional* phrases. We address the tendency of EM to overfit by using Bayesian methods, where sparse priors assign greater mass to parameter vectors with fewer non-zero values therefore favoring shorter, more frequent phrases. We test our model by extracting longer phrases from our model's alignments using traditional phrase extraction, and find that a phrase table based on our system improves MT results over a phrase table extracted from traditional word-level alignments.

2 Phrasal Inversion Transduction Grammar

We use a phrasal extension of Inversion Transduction Grammar (Wu, 1997) as the generative framework. Our ITG has two nonterminals: X and C, where X represents compositional phrase pairs that can have recursive structures and C is the pre-terminal over terminal phrase pairs. There are three rules with X on the left-hand side:

$$\begin{aligned} X &\rightarrow [X\ X], \\ X &\rightarrow \langle X\ X \rangle, \\ X &\rightarrow C. \end{aligned}$$

The first two rules are the straight rule and inverted rule respectively. They split the left-hand side constituent which represents a phrase pair into two smaller phrase pairs on the right-hand side and order them according to one of the two possible permutations. The rewriting process continues until the third rule is invoked. C is our unique pre-terminal for generating terminal multi-word pairs:

$$C \rightarrow \mathbf{e}/\mathbf{f}.$$

We parameterize our probabilistic model in the manner of a PCFG: we associate a multinomial distribution with each nonterminal, where each outcome in this distribution corresponds to an expansion of that nonterminal. Specifically, we place one multinomial distribution θ_X over the three expansions of the nonterminal X, and another multinomial distribution θ_C over the expansions of C. Thus, the parameters in our model can be listed as

$$\theta_{\mathbf{X}} = (P_{\langle\rangle}, P_{[]}, P_C),$$

where $P_{\langle\rangle}$ is for the inverted rule, $P_{[]}$ for the straight rule, P_C for the third rule, satisfying $P_{\langle\rangle} + P_{[]} + P_C = 1$, and

$$\theta_{\mathbf{C}} = (P(\mathbf{e}/\mathbf{f}),\ P(\mathbf{e}'/\mathbf{f}'), \ldots),$$

where $\sum_{\mathbf{e}/\mathbf{f}} P(\mathbf{e}/\mathbf{f}) = 1$ is a multinomial distribution over phrase pairs.

This is our model in a nutshell. We can train this model using a two-dimensional extension of the inside-outside algorithm on bilingual data, assuming every phrase pair that can appear as a leaf in a parse tree of the grammar a valid candidate. However, it is easy to show that the maximum likelihood training will lead to the saturated solution where $P_C = 1$ — each sentence pair is generated by a single phrase spanning the whole sentence. From the computational point of view, the full EM algorithm runs in $O(n^6)$ where n is the average length of the two input sentences, which is too slow in practice.

The key is to control the number of parameters, and therefore the size of the set of candidate phrases. We deal with this problem in two directions. First we change the objective function by incorporating a prior over the phrasal parameters. This has the effect of preferring parameter vectors in $\theta_{\mathbf{C}}$ with fewer non-zero values. Our second approach was to constrain the search space using simpler alignment models, which has the further benefit of significantly speeding up training. First we train a lower level word alignment model, then we place hard constraints on the phrasal alignment space using confident word links from this simpler model. Combining the two approaches, we have a staged training procedure going from the simplest unconstrained word based model to a constrained Bayesian word-level ITG model, and finally proceeding to a constrained Bayesian phrasal model.

3 Variational Bayes for ITG

Goldwater and Griffiths (2007) and Johnson (2007) show that modifying an HMM to include a sparse prior over its parameters and using Bayesian estimation leads to improved accuracy for unsupervised part-of-speech tagging. In this section, we describe a Bayesian estimator for ITG: we select parameters that optimize the probability of the data given a prior. The traditional estimation method for word

alignment models is the EM algorithm (Brown et al., 1993) which iteratively updates parameters to maximize the likelihood of the data. The drawback of maximum likelihood is obvious for phrase-based models. If we do not put any constraint on the distribution of phrases, EM overfits the data by memorizing every sentence pair. A sparse prior over a multinomial distribution such as the distribution of phrase pairs may bias the estimator toward skewed distributions that generalize better. In the context of phrasal models, this means learning the more representative phrases in the space of all possible phrases.

The Dirichlet distribution, which is parameterized by a vector of real values often interpreted as pseudo-counts, is a natural choice for the prior, for two main reasons. First, the Dirichlet is *conjugate* to the multinomial distribution, meaning that if we select a Dirichlet prior and a multinomial likelihood function, the posterior distribution will again be a Dirichlet. This makes parameter estimation quite simple. Second, Dirichlet distributions with small, non-zero parameters place more probability mass on multinomials on the edges or faces of the probability simplex, distributions with fewer non-zero parameters. Starting from the model from Section 2, we propose the following Bayesian extension, where $A \sim \mathrm{Dir}(B)$ means the random variable A is distributed according to a Dirichlet with parameter B:

$$\theta_{\mathbf{X}} \mid \alpha_X \sim \mathrm{Dir}(\alpha_X),$$

$$\theta_{\mathbf{C}} \mid \alpha_C \sim \mathrm{Dir}(\alpha_C),$$

$$\left.\begin{array}{c} [X\ \ X] \\ \langle X\ \ X\rangle \\ C \end{array}\right| X \sim \mathrm{Multi}(\theta_{\mathbf{X}}),$$

$$\mathbf{e}/\mathbf{f} \mid C \sim \mathrm{Multi}(\theta_{\mathbf{C}}).$$

The parameters α_X and α_C control the sparsity of the two distributions in our model. One is the distribution of the three possible branching choices. The other is the distribution of the phrase pairs. α_C is crucial, since the multinomial it is controlling has a high dimension. By adjusting α_C to a very small number, we hope to place more posterior mass on parsimonious solutions with fewer but more confident and general phrase pairs.

Having defined the Bayesian model, it remains to decide the inference procedure. We chose Variational Bayes, for its procedural similarity to EM and ease of implementation. Another potential option would be Gibbs sampling (or some other sampling technique). However, in experiments in unsupervised POS tag learning using HMM structured models, Johnson (2007) shows that VB is more effective than Gibbs sampling in approaching distributions that agree with the Zipf's law, which is prominent in natural languages.

Kurihara and Sato (2006) describe VB for PCFGs, showing the only need is to change the M step of the EM algorithm. As in the case of maximum likelihood estimation, Bayesian estimation for ITGs is very similar to PCFGs, which follows due to the strong isomorphism between the two models. Specific to our ITG case, the M step becomes:

$$\tilde{P}_{[]}^{(l+1)} = \frac{\exp(\psi(E(X \to [X\ X]) + \alpha_X))}{\exp(\psi(E(X) + s\alpha_X))},$$

$$\tilde{P}_{\langle\rangle}^{(l+1)} = \frac{\exp(\psi(E(X \to \langle X\ X\rangle) + \alpha_X))}{\exp(\psi(E(X) + s\alpha_X))},$$

$$\tilde{P}_{C}^{(l+1)} = \frac{\exp(\psi(E(X \to C) + \alpha_X))}{\exp(\psi(E(X) + s\alpha_X))},$$

$$\tilde{P}^{(l+1)}(\mathbf{e}/\mathbf{f}) = \frac{\exp(\psi(E(\mathbf{e}/\mathbf{f}) + \alpha_C))}{\exp(\psi(E(C) + m\alpha_C))},$$

where ψ is the *digamma function* (Beal, 2003), $s = 3$ is the number of right-hand-sides for X, and m is the number of observed phrase pairs in the data. The sole difference between EM and VB with a sparse prior α is that the raw fractional counts c are replaced by $\exp(\psi(c + \alpha))$, an operation that resembles smoothing. As pointed out by Johnson (2007), in effect this expression adds to c a small value that asymptotically approaches $\alpha - 0.5$ as c approaches ∞, and 0 as c approaches 0. For small values of α the net effect is the opposite of typical smoothing, since it tends to redistribute probably mass away from unlikely events onto more likely ones.

4 Bitext Pruning Strategy

ITG is slow mainly because it considers every pair of spans in two sentences as a possible chart element. In reality, the set of useful chart elements is much

smaller than the possible $scriptO(n^4)$, where n is the average sentence length. Pruning the span pairs (bitext cells) that can participate in a tree (either as terminals or non-terminals) serves to not only speed up ITG parsing, but also to provide a kind of initialization hint to the training procedures, encouraging it to focus on promising regions of the alignment space.

Given a bitext cell defined by the four boundary indices (i, j, l, m) as shown in Figure 1a, we prune based on a figure of merit $V(i, j, l, m)$ approximating the utility of that cell in a full ITG parse. The figure of merit considers the Model 1 scores of not only the words inside a given cell, but also all the words not included in the source and target spans, as in Moore (2003) and Vogel (2005). Like Zhang and Gildea (2005), it is used to prune bitext cells rather than score phrases. The total score is the product of the Model 1 probabilities for each column; "inside" columns in the range $[l, m]$ are scored according to the sum (or maximum) of Model 1 probabilities for $[i, j]$, and "outside" columns use the sum (or maximum) of all probabilities not in the range $[i, j]$.

Our pruning differs from Zhang and Gildea (2005) in two major ways. First, we perform pruning using both directions of the IBM Model 1 scores; instead of a single figure of merit V, we have two: V_F and V_B. Only those spans that pass the pruning threshold in both directions are kept. Second, we allow whole spans to be pruned. The figure of merit for a span is $V_F(i, j) = \max_{l,m} V_F(i, j, l, m)$. Only spans that are within some threshold of the unrestricted Model 1 scores V_F and V_B are kept:

$$\frac{V_F(i,j)}{V_F} \geq \tau_s \text{ and } \frac{V_B(l,m)}{V_B} \geq \tau_s.$$

Amongst those spans retained by this first threshold, we keep only those bitext cells satisfying both

$$\frac{V_F(i,j,l,m)}{V_F(i,j)} \geq \tau_b \text{ and } \frac{V_B(i,j,l,m)}{V_B(l,m)} \geq \tau_b.$$

4.1 Fast Tic-tac-toe Pruning

The tic-tac-toe pruning algorithm (Zhang and Gildea, 2005) uses dynamic programming to compute the product of inside and outside scores for all cells in $\mathcal{O}(n^4)$ time. However, even this can be slow for large values of n. Therefore we describe an improved algorithm with best case n^3 performance. Although the worst case performance is also $\mathcal{O}(n^4)$, in practice it is significantly faster.

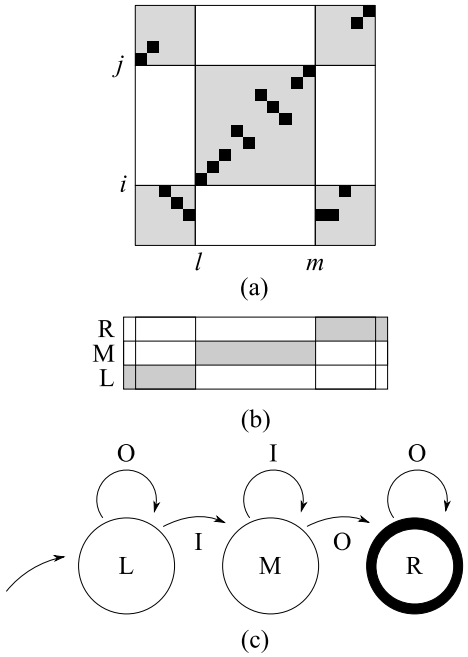

Figure 1: (a) shows the original tic-tac-toe score for a bitext cell (i, j, l, m). (b) demonstrates the finite state representation using the machine in (c), assuming a fixed source span (i, j).

To begin, let us restrict our attention to the forward direction for a fixed source span (i, j). Pruning bitext spans and cells requires $V_F(i, j)$, the score of the best bitext cell within a given span, as well as all cells within a given threshold of that best score. For a fixed i and j, we need to search over the starting and ending points l and m of the inside region. Note that there is an isomorphism between the set of spans and a simple finite state machine: any span (l, m) can be represented by a sequence of l OUTSIDE columns, followed by $m-l+1$ INSIDE columns, followed by $n - m + 1$ OUTSIDE columns. This simple machine has the restricted form described in Figure 1c: it has three states, L, M, and R; each transition generates either an OUTSIDE column O or an INSIDE column I. The cost of generating an OUTSIDE at position a is $O(a) = P(t_a|\text{NULL}) + \sum_{b \notin [i,j]} P(t_a|s_b)$; likewise the cost of generating an INSIDE column is $I(a) = P(t_a|\text{NULL}) + \sum_{b \in [i,j]} P(t_a|s_b)$, with

$O(0) = O(n+1) = 1$ and $I(0) = I(n+1) = 0$.

Directly computing O and I would take time $\mathcal{O}(n^2)$ for each source span, leading to an overall runtime of $\mathcal{O}(n^4)$. Luckily there are faster ways to find the inside and outside scores. First we can precompute following arrays in $\mathcal{O}(n^2)$ time and space:

$$\begin{aligned} \text{pre}[0,l] &:= P(t_l|\text{NULL}) \\ \text{pre}[i,l] &:= \text{pre}[i-1,l] + P(t_l|s_i) \\ \text{suf}[n+1,l] &:= 0 \\ \text{suf}[i,l] &:= \text{suf}[i+1,l] + P(t_l|s_i) \end{aligned}$$

Then for any (i,j), $O(a) = P(t_a|\text{NULL}) + \sum_{b \notin [i,j]} P(t_a|s_b) = \text{pre}[i-1,a] + \text{suf}[j+1,a]$. $I(a)$ can be incrementally updated as the source span varies: when $i = j$, $I(a) = P(t_a|\text{NULL}) + P(t_a|s_i)$. As j is incremented, we add $P(t_a|s_j)$ to $I(a)$. Thus we have linear time updates for O and I.

We can then find the best scoring sequence using the familiar Viterbi algorithm. Let $\delta[a,\sigma]$ be the cost of the best scoring sequence ending at in state σ at time a:

$$\begin{aligned} \delta[0,\sigma] &:= 1 \text{ if } \sigma = L\text{; } 0 \text{ otherwise} \\ \delta[a,L] &:= \delta[a-1,L] \cdot O(a) \\ \delta[a,M] &:= \max_{\sigma \in L,M} \{\delta[a-1,\sigma]\} \cdot I(a) \\ \delta[a,R] &:= \max_{\sigma \in M,R} \{\delta[a-1,\sigma]\} \cdot O(a) \end{aligned}$$

Then $V_F(i,j) = \delta[n+1,R]$, using the isomorphism between state sequences and spans. This linear time algorithm allows us to compute span pruning in $\mathcal{O}(n^3)$ time. The same algorithm may be performed using the backward figure of merit after transposing rows and columns.

Having cast the problem in terms of finite state automata, we can use finite state algorithms for pruning. For instance, fixing a source span we can enumerate the target spans in decreasing order by score (Soong and Huang, 1991), stopping once we encounter the first span below threshold. In practice the overhead of maintaining the priority queue outweighs any benefit, as seen in Figure 2.

An alternate approach that avoids this overhead is to enumerate spans by position. Note that $\delta[m,R] \cdot \prod_{a=m+1}^{n} O(a)$ is within threshold iff there is a span with right boundary $m' < m$ within threshold. Furthermore if $\delta[m,M] \cdot \prod_{a=m+1}^{n} O(a)$ is within threshold, then m is the right boundary within threshold. Using these facts, we can gradually sweep the right boundary m from n toward 1 until the first condition fails to hold. For each value where the second condition holds, we pause to search for the set of left boundaries within threshold.

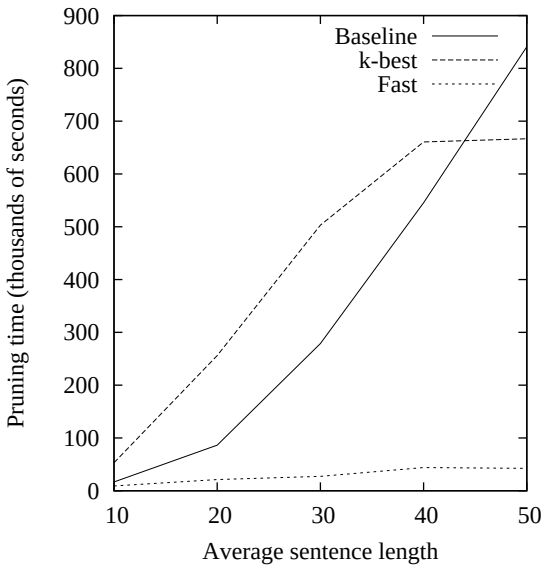

Figure 2: Speed comparison of the $O(n^4)$ tic-tac-toe pruning algorithm, the A* top-x algorithm, and the fast tic-tac-toe pruning. All produce the same set of bitext cells, those within threshold of the best bitext cell.

Likewise for the left edge, $\delta[l,M] \cdot \prod_{a=l+1}^{m} I(a) \cdot \prod_{a=m+1}^{n} O(a)$ is within threshold iff there is some $l' < l$ identifying a span (l',m) within threshold. Finally if $V(i,j,l,m) = \delta[l-1,L] \cdot \prod_{a=l}^{m} I(a) \cdot \prod_{a=m+1}^{n} O(a)$ is within threshold, then (i,j,l,m) is a bitext cell within threshold. For right edges that are known to be within threshold, we can sweep the left edges leftward until the first condition no longer holds, keeping only those spans for which the second condition holds.

The filtering algorithm behaves extremely well. Although the worst case runtime is still $\mathcal{O}(n^4)$, the best case has improved to n^3; empirically it seems to significantly reduce the amount of time spent exploring spans. Figure 2 compares the speed of the fast tic-tac-toe algorithm against the algorithm in Zhang and Gildea (2005).

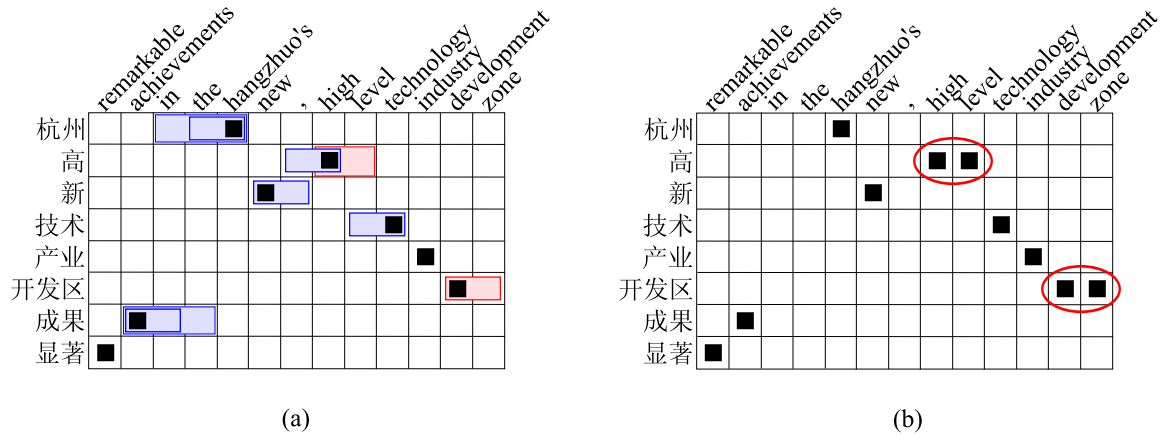

Figure 3: Example output from the ITG using non-compositional phrases. (a) is the Viterbi alignment from the word-based ITG. The shaded regions indicate phrasal alignments that are allowed by the non-compositional constraint; all other phrasal alignments will not be considered. (b) is the Viterbi alignment from the phrasal ITG, with the multi-word alignments highlighted.

5 Bootstrapping Phrasal ITG from Word-based ITG

This section introduces a technique that bootstraps candidate phrase pairs for phrase-based ITG from word-based ITG Viterbi alignments. The word-based ITG uses the same expansions for the non-terminal X, but the expansions of C are limited to generate only 1-1, 1-0, and 0-1 alignments:

$$\begin{aligned} C &\rightarrow e/f, \\ C &\rightarrow e/\epsilon, \\ C &\rightarrow \epsilon/f \end{aligned}$$

where ϵ indicates that no word was generated. Broadly speaking, the goal of this section is the same as the previous section, namely, to limit the set of phrase pairs that needs to be considered in the training process. The tic-tac-toe pruning relies on IBM model 1 for scoring a given aligned area. In this part, we use word-based ITG alignments as anchor points in the alignment space to pin down the potential phrases. The scope of iterative phrasal ITG training, therefore, is limited to determining the boundaries of the phrases anchored on the given one-to-one word alignments.

The heuristic method is based on the Non-Compositional Constraint of Cherry and Lin (2007). Cherry and Lin (2007) use GIZA++ intersections which have high precision as anchor points in the bitext space to constraint ITG phrases. We use ITG Viterbi alignments instead. The benefit is two-fold. First of all, we do not have to run a GIZA++ aligner. Second, we do not need to worry about non-ITG word alignments, such as the $(2, 4, 1, 3)$ permutation patterns. GIZA++ does not limit the set of permutations allowed during translation, so it can produce permutations that are not reachable using an ITG.

Formally, given a word-based ITG alignment, the bootstrapping algorithm finds all the phrase pairs according to the definition of Och and Ney (2004) and Chiang (2005) with the additional constraint that each phrase pair contains at most one word link. Mathematically, let $e(i, j)$ count the number of word links that are emitted from the substring $\mathbf{e}_{i...j}$, and $f(l, m)$ count the number of word links emitted from the substring $\mathbf{f}_{l...m}$. The non-compositional phrase pairs satisfy

$$e(i, j) = f(l, m) \leq 1.$$

Figure 3 (a) shows all possible non-compositional phrases given the Viterbi word alignment of the example sentence pair.

6 Summary of the Pipeline

We summarize the pipeline of our system, demonstrating the interactions between the three main contributions of this paper: Variational Bayes, tic-tac-toe pruning, and word-to-phrase bootstrapping. We

start from sentence-aligned bilingual data and run IBM Model 1 in both directions to obtain two translation tables. Then we use the efficient bidirectional tic-tac-toe pruning to prune the bitext space within each of the sentence pairs; ITG parsing will be carried out on only this this sparse set of bitext cells. The first stage of training is word-based ITG, using the standard iterative training procedure, except VB replaces EM to focus on a sparse prior. After several training iterations, we obtain the Viterbi alignments on the training data according to the final model. Now we transition into the second stage – the phrasal training. Before the training starts, we apply the non-compositional constraints over the pruned bitext space to further constrain the space of phrase pairs. Finally, we run phrasal ITG iterative training using VB for a certain number of iterations. In the end, a Viterbi pass for the phrasal ITG is executed to produce the non-compositional phrasal alignments. From this alignment, phrase pairs are extracted in the usual manner, and a phrase-based translation system is trained.

7 Experiments

The training data was a subset of 175K sentence pairs from the NIST Chinese-English training data, automatically selected to maximize character-level overlap with the source side of the test data. We put a length limit of 35 on both sides, producing a training set of 141K sentence pairs. 500 Chinese-English pairs from this set were manually aligned and used as a gold standard.

7.1 Word Alignment Evaluation

First, using evaluations of alignment quality, we demonstrate the effectiveness of VB over EM, and explore the effect of the prior.

Figure 4 examines the difference between EM and VB with varying sparse priors for the word-based model of ITG on the 500 sentence pairs, both after 10 iterations of training. Using EM, because of overfitting, AER drops first and increases again as the number of iterations varies from 1 to 10. The lowest AER using EM is achieved after the second iteration, which is .40. At iteration 10, AER for EM increases to .42. On the other hand, using VB, AER decreases monotonically over the 10 iterations and

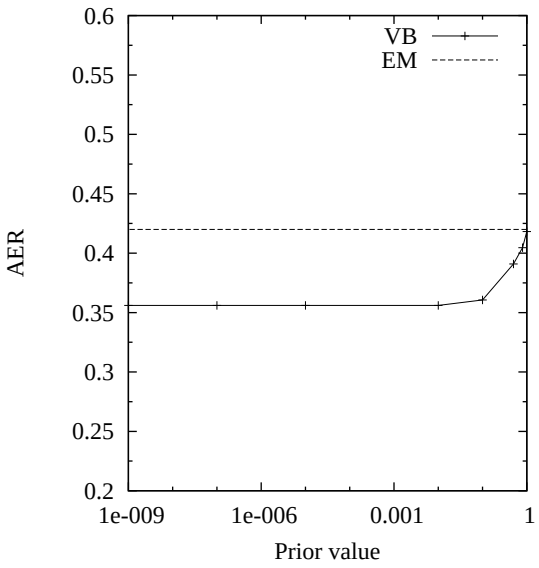

Figure 4: AER drops as α_C approaches zero; a more sparse solution leads to better results.

stabilizes at iteration 10. When α_C is $1e-9$, VB gets AER close to .35 at iteration 10.

As we increase the bias toward sparsity, the AER decreases, following a long slow plateau. Although the magnitude of improvement is not large, the trend is encouraging.

These experiments also indicate that a very sparse prior is needed for machine translation tasks. Unlike Johnson (2007), who found optimal performance when α was approximately 10^{-4}, we observed monotonic increases in performance as α dropped. The dimensionality of this MT problem is significantly larger than that of the sequence problem, though, therefore it may take a stronger push from the prior to achieve the desired result.

7.2 End-to-end Evaluation

Given an unlimited amount of time, we would tune the prior to maximize end-to-end performance, using an objective function such as BLEU. Unfortunately these experiments are very slow. Since we observed monotonic increases in alignment performance with smaller values of α_C, we simply fixed the prior at a very small value (10^{-100}) for all translation experiments. We do compare VB against EM in terms of final BLEU scores in the translation experiments to ensure that this sparse prior has a sig-

nificant impact on the output.

We also trained a baseline model with GIZA++ (Och and Ney, 2003) following a regimen of 5 iterations of Model 1, 5 iterations of HMM, and 5 iterations of Model 4. We computed Chinese-to-English and English-to-Chinese word translation tables using five iterations of Model 1. These values were used to perform tic-tac-toe pruning with $\tau_b = 1 \times 10^{-3}$ and $\tau_s = 1 \times 10^{-6}$. Over the pruned charts, we ran 10 iterations of word-based ITG using EM or VB. The charts were then pruned further by applying the non-compositional constraint from the Viterbi alignment links of that model. Finally we ran 10 iterations of phrase-based ITG over the residual charts, using EM or VB, and extracted the Viterbi alignments.

For translation, we used the standard phrasal decoding approach, based on a re-implementation of the Pharaoh system (Koehn, 2004). The output of the word alignment systems (GIZA++ or ITG) were fed to a standard phrase extraction procedure that extracted all phrases of length up to 7 and estimated the conditional probabilities of source given target and target given source using relative frequencies. Thus our phrasal ITG learns only the minimal non-compositional phrases; the standard phrase-extraction algorithm learns larger combinations of these minimal units. In addition the phrases were annotated with lexical weights using the IBM Model 1 tables. The decoder also used a trigram language model trained on the target side of the training data, as well as word count, phrase count, and distortion penalty features. Minimum Error Rate training (Och, 2003) over BLEU was used to optimize the weights for each of these models over the development test data.

We used the NIST 2002 evaluation datasets for tuning and evaluation; the 10-reference development set was used for minimum error rate training, and the 4-reference test set was used for evaluation. We trained several phrasal translation systems, varying only the word alignment (or phrasal alignment) method.

Table 1 compares the four systems: the GIZA++ baseline, the ITG word-based model, the ITG multi-word model using EM training, and the ITG multi-word model using VB training. ITG-mwm-VB is our best model. We see an improvement of nearly 2 points dev set and nearly 1 point of improvement on the test set. We also observe the consistent superiority of VB over EM. The gain is especially large on the test data set, indicating VB is less prone to overfitting.

	Development	*Test*
GIZA++	37.46	28.24
ITG-word	35.47	26.55
ITG-mwm (VB)	39.21	**29.02**
ITG-mwm (EM)	39.15	28.47

Table 1: Translation results on Chinese-English, using the subset of training data (141K sentence pairs) that have length limit 35 on both sides. (No length limit in translation.)

8 Conclusion

We have presented an improved and more efficient method of estimating phrase pairs directly. By both changing the objective function to include a bias toward sparser models and improving the pruning techniques and efficiency, we achieve significant gains on test data with practical speed. In addition, these gains were shown without resorting to external models, such as GIZA++. We have shown that VB is both practical and effective for use in MT models.

However, our best system does not apply VB to a single probability model, as we found an appreciable benefit from bootstrapping each model from simpler models, much as the IBM word alignment models are usually trained in succession. We find that VB alone is not sufficient to counteract the tendency of EM to prefer analyses with smaller trees using fewer rules and longer phrases. Both the tic-tac-toe pruning and the non-compositional constraint address this problem by reducing the space of possible phrase pairs. On top of these hard constraints, the sparse prior of VB helps make the model less prone to overfitting to infrequent phrase pairs, and thus improves the quality of the phrase pairs the model learns.

Acknowledgments This work was done while the first author was at Microsoft Research; thanks to Xiaodong He, Mark Johnson, and Kristina Toutanova. The last author was supported by NSF IIS-0546554.

References

Matthew Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June.

Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 17–24, Rochester, New York, April. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, Michigan, USA.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June. Association for Computational Linguistics.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 115–124, Washington, USA, September.

Kenichi Kurihara and Taisuke Sato. 2006. Variational bayesian grammar induction for natural language. In *International Colloquium on Grammatical Inference*, pages 84–96, Tokyo, Japan.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Robert C. Moore. 2003. Learning translations of named-entity phrases from parallel corpora. In *Proceedings of EACL*, Budapest, Hungary.

Franz Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, December.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167, Sapporo, Japan.

Frank Soong and Eng Huang. 1991. A tree-trellis based fast search for finding the n best sentence hypotheses in continuous speech recognition. In *Proceedings of ICASSP 1991*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING*, pages 836–741, Copenhagen, Denmark.

Stephan Vogel. 2005. PESA: Phrase pair extraction as sentence splitting. In *MT Summit X*, Phuket, Thailand.

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

Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of ACL*.