Improving the Performance of GIZA++ Using Variational Bayes

Darcey Riley Daniel Gildea

The University of Rochester
Computer Science Department
Rochester, NY 14627

Technical Report 963

December 2010

Abstract

Bayesian approaches have been shown to reduce the amount of overfitting that occurs when running the EM algorithm, by placing prior probabilities on the model parameters. We apply one such Bayesian technique, variational Bayes, to GIZA++, a widely-used piece of software that computes word alignments for statistical machine translation. We show that using variational Bayes improves the performance of GIZA++, as well as improving the overall performance of the Moses machine translation system in terms of BLEU score.

This work was supported by NSF grants IIS-0546554 and IIS-0910611.

1 Introduction

GIZA++ (Och and Ney, 2003) is a program that trains the IBM Models (Brown et al., 1993) as well as a Hidden Markov Model (HMM) (Vogel et al., 1996), and uses these models to compute Viterbi alignments for statistical machine translation. While GIZA++ can be used on its own, it typically serves as the starting point for other machine translation systems, both phrase-based and syntactic. For instance, running GIZA++ is the first step in training the popular phrase-based translation system Moses (Koehn et al., 2007). The hierarchical phrase-based translation system Hiero (Chiang, 2005) also uses GIZA++ to generate word alignments. Galley et al. (2004) use word alignments from GIZA++ to learn rules for syntax-based machine translation.

Both the IBM Models and the Hidden Markov Model are trained using the EM algorithm. Because EM chooses parameters which maximize the likelihood of the data, it tends to overfit the parameters to the data. Johnson (2007) showed that Hidden Markov Models for part-of-speech tagging perform better when the number of hidden states is restricted; when more hidden states are allowed, the forward-backward algorithm (Baum, 1972), a version of the EM algorithm (Dempster et al., 1977) which trains HMMs, will overfit the model to the data. Johnson experimented with Bayesian techniques, which use a prior on the parameters to discourage them from taking on unreasonable values, and found that using variational Bayes decreased the amount of overfitting that occurred when more hidden states were used. Bayesian techniques in general and variational Bayes in particular have been used to control overfitting in a number of natural language processing applications. In machine translation, Blunsom et al. (2008) and DeNero et al. (2008) use Bayesian techniques to learn bilingual phrase pairs. In this setting, which involves finding a segmentation of the input sentences into phrasal units, it is particularly important to control the tendency of EM to choose longer phrases, which explain the training data well but are unlikely to generalize. This report, in contrast, is concerned with word-level translation models. Since word-level alignments are widely used as the first step in most current translation systems, our hope is that improvements in this component of the machine translation pipeline will be of broad utility to other researchers in the area.

Moore (2004) discusses details of how EM overfits the data when training IBM Model 1. He discovered that the EM algorithm is particularly susceptible to overfitting in the case of rare words, for an interesting reason. Suppose a sentence contains an English word e_1 that occurs nowhere else in the data and its French translation f_1. Suppose that same sentence also contains a word e_2 which occurs frequently in the overall data but whose translation in this sentence, f_2, co-occurs with it infrequently. If the translation $t(f_2|e_2)$ occurs with probability 0.1, then the sentence will have a higher probability if EM assigns the rare word and its actual translation a probability of $t(f_1|e_1) = 0.5$, and assigns the rare word's translation to f_2 a probability of $t(f_2|e_1) = 0.5$, than if it assigns a probability of 1 to the correct translation $t(f_1|e_1)$. Moore suggests a number of solutions to this issue, including add-n smoothing and initializing the probabilities based on a heuristic rather than choosing uniform probabilities. When combined, his solutions cause a significant decrease in alignment error rate (AER).

Like Moore, we seek to improve the performance of the IBM Models. Like Johnson, we use a variational Bayesian version of the EM algorithm. For the translation probabilities $t(f|e)$, variational Bayes performs a kind of "anti-smoothing" that helps to control overfitting.

In this technical report, we describe the variational Bayes techniques and how they can be used

to control overfitting and improve the performance of the IBM Models. We explain our modifications to GIZA++, and then present our results in terms of perplexity of held-out test data, alignment error rate (AER), and the BLEU scores of an end-to-end phrase-based machine translation system. We show an improvement to BLEU score when using our version of GIZA++ in the Moses system. Finally, we explain how our modifications to GIZA++ can be used.

2 Bayesian Models and Variational Bayes

The goal of the EM algorithm is to determine the parameters θ which maximize the likelihood of the observed data $\mathbf{X}$, taking into account all the possible values of the hidden variables $\mathbf{Z}$:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, p(\mathbf{X}|\theta) \tag{1}$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta) \tag{2}$$

For a detailed explanation of the EM algorithm, we refer the reader to Dempster et al. (1977) or Beal (2003).

Such a framework allows the EM algorithm to choose whichever parameters do the best job of fitting the data. For this reason, EM often suffers from overfitting, especially if there are a large number of parameters compared to the amount of data. This is the problem that Johnson (2007) discovered when using Hidden Markov Models for part-of-speech tagging: when more hidden states were allowed, the HMM was more subject to overfitting. An HMM that was allowed to use as many hidden states as it liked could assign a different hidden state to each word, giving the data a probability of one but creating an entirely unrealistic model.

Additionally, we often have some idea of what sort of values the parameters should take. For instance, Johnson notes that in part-of-speech tagging, we would expect some part-of-speech tags to occur very frequently and others to occur rarely, but the EM algorithm does not assign such a skewed distribution to the data. Similarly, for machine translation, we would expect the probability distribution for translations to resemble Zipf's law, but the EM algorithm assigns the data a much flatter distribution.

2.1 Bayesian Models

To remedy these issues, we can take a Bayesian stance and say that the parameters themselves have been drawn from some probability distribution. The parameters of this extra probability distribution are called hyperparameters, and are denoted by α. By choosing their values appropriately, we can bias the model towards learning the sort of parameters we would like it to learn.

The goal of Bayesian methods is to compute the likelihood of a new data point x^{n+1} given data points x^1 through x^n that have already been seen, and given the hyperparameters. This involves integrating over the model parameters and the hidden states, both of which are now encompassed by θ. They have been combined into one variable because in a Bayesian model, the parameters act

as hidden variables: they, like the hidden variables, are random variables that have been drawn from a probability distribution.

$$p(x^{n+1}|x^1,\ldots,x^n,\alpha) = \int p(x^{n+1}|x^1,\ldots,x^n,\theta)p(\theta|\alpha)d\theta \tag{3}$$

Because the parameters θ for the IBM Models and the HMM can be viewed mathematically as multinomials, the priors over θ for these and similar models are typically chosen to be Dirichlet distributions, which are conjugate to multinomial distributions. It is important to choose a conjugate prior so that inference will be tractable.

When using this Bayesian framework with the IBM Models and HMM, we do not have any prior data, and so we assume our data is the first data point. The hyperparameters represent our prior knowledge about the data and how certain we are of it. In practice, it acts like imaginary data which gets added into the counts during normalization. Usually, normalization without hidden variables works like this:

$$\theta_{x_i|y} = \frac{c(x_i|y)}{\sum_j c(x_j|y)} \tag{4}$$

Here, the count $c(x|y)$ denotes the number of times that x and y appear together. With the Dirichlet priors, however, the formula for normalization becomes

$$\theta_{x_i|y} = \frac{c(x_i|y) + \alpha_i}{\sum_j c(x_j|y) + \alpha_j} \tag{5}$$

where the α_js are the elements of the hyperparameter vector. Thus, based on the choice of α, we can bias the model towards a certain conclusion *a priori*. Higher values of α indicate more certainty in our prior knowledge, and influence the counts more.

We use a symmetric Dirichlet prior, i.e., one for which all of the αs are the same. This means that the prior will act as a form of add-n smoothing, where $\alpha = n$.

2.2 Bayesian Models and EM

Thus, there are many benefits to be gained from introducing these Dirichlet priors into the model. Unfortunately, adding a prior over the parameters means that we cannot use EM directly. This is because the EM algorithm makes implicit independence assumptions. First it holds the parameters fixed and optimizes the hidden variables with respect to them; then it holds the hidden variables fixed and optimizes the parameters. EM algorithms typically rely on dynamic programming to compute expected counts for hidden variables.

These dynamic programming algorithms are possible because, for a non-Bayesian model, the parameters are treated as fixed quantities rather than random variables. Thus the probabilities of the hidden variables decompose according to such a model, and there are no independence assumptions required for the E step. For the M step, we make the assumption that the parameters can be optimized independently of their effects on the hidden variables, and it is this assumption that makes the EM algorithm an approximate algorithm which converges on the solution rather than finding it directly.

For the Bayesian model, on the other hand, the parameters are random variables, and thus they are interdependent with the hidden variables; both are conditioned on the fixed hyperparameter α. This makes it impossible to compute exact expected counts for all the hidden variables with respect to the parameters during the E step, and this is why EM cannot be used with Bayesian models.

Thus, if we want to use the Bayesian model, we must resort to approximate algorithms. Johnson (2007) tested two such approximate Bayesian algorithms, Gibbs sampling and variational Bayes, and found variational Bayes to be superior.

2.3 Variational Bayes

Variational Bayes closely resembles the usual form of the EM algorithm. Like normal EM, it makes the assumption that the parameters and hidden variables are independent, and is thus an approximation of the Bayesian model. Variational Bayes is frequently described as an "approximate algorithm", and EM as an "exact algorithm". However, variational Bayes is really no more approximate than EM. It is true that variational Bayes requires an approximation of its model while EM does not; however, the original EM algorithm is, in a sense, already approximate, because while each update is computed exactly, these updates only approximate the global optimum.

Beal (Beal, 2003) gives a detailed derivation of a variational Bayesian algorithm for HMMs. The result is a very slight change to the M step of the original EM algorithm. During the M step of the original EM algorithm, the expected counts collected in the E step are normalized to give the new values of the parameters:

$$\theta_{x_i|y} = \frac{E[c(x_i|y)]}{\sum_j E[c(x_j|y)]} \tag{6}$$

The variational Bayesian M step performs an inexact normalization, where the resulting parameters will add up to less than one, by passing the expected counts collected in the E step through the function $f(v) = \exp(\psi(v))$, where ψ is the digamma function (Johnson, 2007):

$$\theta_{x_i|y} = \frac{f(E[c(x_i|y)] + \alpha)}{f(\sum_j (E[c(x_j|y)] + \alpha))} \tag{7}$$

This modified M step can be applied to the IBM Models as well as HMMs, and is thus what we use for GIZA++.

In practice, the digamma function has the effect of subtracting 0.5 from its argument. When α is set to a low value, this has the effect of "anti-smoothing". For the translation probabilities, because about 0.5 is subtracted from the expected counts, small counts corresponding to rare co-occurrences of words will be penalized heavily, while larger counts will not be affected very much. Thus, low values of α cause the algorithm to favor words which co-occur frequently and to distrust words that co-occur rarely. In this way, variational Bayes controls the overfitting that would otherwise occur with rare words. On the other hand, higher values of α can be chosen if smoothing is desired, for instance in the case of the alignment probabilities, which state how likely a word in position i of the English sentence is to align to a word in position j of the French sentence. For these probabilities, smoothing is important because we do not want to rule out any alignment altogether, no matter how infrequently it occurs in the data.

3 Modifying GIZA++

In accordance with the equations shown above, we have modified the M step of the EM algorithm in GIZA++ to use variational Bayes.

GIZA++ stores its probability distributions in tables, keeping separate tables for translation probabilities, alignment probabilities, and the other probability distributions that it uses. We have modified the code for the translation and alignment probability tables, as the M step of the EM algorithm is performed by normalization functions in these classes. All of the IBM Models, along with the HMM, use the translation probability table; other probability tables are added when moving to the later IBM Models and HMM.

There are actually two versions of the class which stores the translation probabilities; the flags that GIZA++ is compiled with determine which version will be used. One version stores the probabilities in a hashmap, while the other version uses a vector of vectors in order to save memory, and accesses its probabilities using binary search. There are many subtle differences between these two versions, some of which affect the way normalization is done.

The binary search version of the code normalizes straightforwardly and does not require any further explanation. The hashmap version, on the other hand, performs a more complicated normalization. After normalizing, it removes all probabilities below a certain threshold, then normalizes again, then repeats this process. The number of iterative normalizations used are specified by the calling function.

This iterative normalization creates a number of problems for variational Bayes, and must be turned off. One issue is the fact that normalization using variational Bayes will give probabilities that add up to less than one. Thus, some possible translations in the table will be pruned unnecessarily. The main problem, however, lies in the fact that variational Bayes subtracts about 0.5 from each of the counts. On the first normalization, for common pairs of words these counts will be high, and the effect will be small. However, the results of normalization using variational Bayes are fractional, and these will be stored as the counts to be used during the next normalization. When these fractions are passed into the f function, the numerators will become disproportionately small compared to the denominators, and the resulting fractions will be tiny. Many of these will be pruned out. After just a few iterative normalizations, most of the possible translations will have been pruned out of the table. Thus, when the user selects variational Bayes as an option, we do not allow the translation tables to iteratively normalize.

In most of our experiments, we do perform a single additional round of normalization without variational Bayes, for both versions of the translation tables, to make the translation probabilities add up to one. We make sure that nothing is pruned from the hashmap version of the translation tables until this normalization has completed, thereby solving the first issue discussed above. Our reason for implementing this additional normalization, however, was so that we could calculate perplexities comparable to the original IBM Models, which requires a distribution that sums to one. Normalizing the probabilities back to one actually detracts from the effect of variational Bayes, giving an algorithm that overfits less than the original EM algorithm but more than variational Bayes. In practice, however, we found that this extra round of normalization had almost no effect on the AER.

We also implemented variational Bayes for the alignment tables, though we discovered that including variational Bayes for these probabilities had relatively little affect. The alignment table

Sentence pair	count
e_2 f_3	9
e_2 f_2	2
e_1 e_2 f_1 f_2	1

Table 1: An example of data with rare words.

in its original form does some smoothing during normalization by interpolating the counts with a uniform distribution. Because variational Bayes can itself be a form of smoothing, we remove this smoothing when variational Bayes is used with the alignment tables. Choosing higher values of α will cause more smoothing to occur.

4 Results

First, we ran our modified version of GIZA++ on a simple test case designed to be similar to the example of Moore (2004) which we discussed in Section 1. Our test case, shown in Table 1, had three different sentence pairs; we included nine instances of the first, two instances of the second, and one of the third.

Human intuition tells us that f_2 should translate to e_2 and f_1 should translate to e_1. However, the EM algorithm without variational Bayes prefers e_1 as the translation of f_2, for the reasons discussed above. The EM algorithm with variational Bayes does not overfit this data and prefers e_2 as f_2's translation, both with and without the extra normalization discussed in the previous section.

For our experiments with bilingual data, we used three language pairs: French and English, Chinese and English, and German and English. We used Canadian Hansard data for French-English, Europarl data for German-English, and newswire data for Chinese-English. For measuring alignment error rate, we used 447 French-English sentences provided by Hermann Ney and Franz Och containing both sure and possible alignments, while for German-English we used 220 sentences provided by Chris Callison-Burch with sure alignments only, and for Chinese-English we used the first 400 sentences of the data provided by Yang Liu, also with sure alignments only. For computing BLEU scores, we used single reference datasets for French-English and German-English, and four references for Chinese-English. For minimum error rate training, we used 1000 sentences for French-English, 2000 sentences for German-English, and 1274 sentences for Chinese-English. Our test sets contained 1000 sentences each for French-English and German-English, and 686 sentences for Chinese-English. 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).

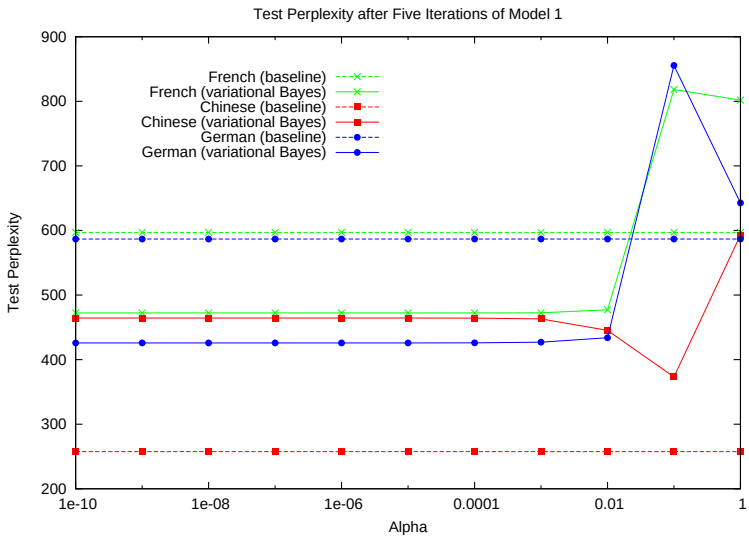

Figure 1: Determining the best value of α for the translation probabilities. Training data is 10000 sentence pairs from each language pair. Variational Bayes is used for Model 1 only. This table shows the test perplexity for different values of α after Model 1 has been run for five iterations.

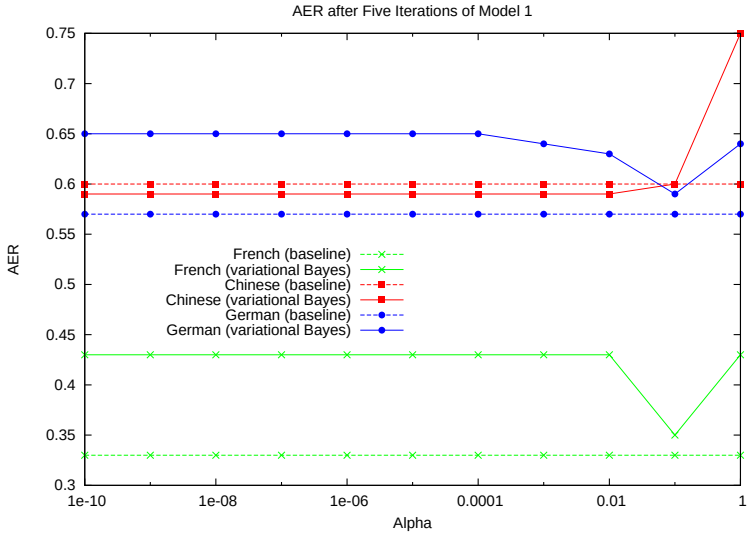

Figure 2: Determining the best value of α for the translation probabilities. Training data is 10000 sentence pairs from each language pair. Variational Bayes is used for Model 1 only. This table shows the AER for different values of α after Model 1 has been run for five iterations.

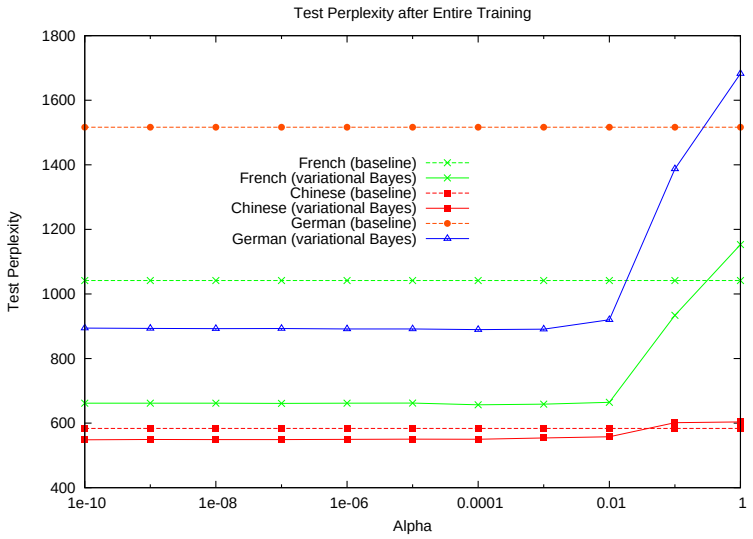

Figure 3: Determining the best value of α for the translation probabilities. Training data is 10000 sentence pairs from each language pair. Variational Bayes is used for Model 1 only. This table shows the test perplexity for different values of α after training is complete (five iterations each of Models 1, HMM, 3, and 4).

We ran our code on ten thousand sentence pairs to determine the best value of α for the translation probabilities $t(f|e)$. For our training, we ran GIZA++ for five iterations each of Model 1, the HMM, Model 3, and Model 4. Variational Bayes was only used for Model 1. Figures 1 through 4 show how variational Bayes in general, and different values of α in particular, affect the performance of GIZA++, in terms of test perplexity and AER. We find that low values of α cause the most consistent improvements, and so we use $\alpha = 0$ for the translation probabilities in the remaining experiments. Note that, while a value of $\alpha = 0$ does not define a probabilistically valid Dirichlet prior, it does not cause any practical problems in the update equation for variational Bayes.

After five iterations of Model 1, we discover that although variational Bayes causes some improvements for all three language pairs, it raises the AER (Figure 2) for French and German, and raises the test perplexity (Figure 1) for Chinese. However, after training is complete, it is clear that variational Bayes has improved the performance of the overall system, lowering both test perplexity (Figure 3) and AER (Figure 4) for all three language pairs. Because Model 1 is typically used to initialize the later models, we are not nearly as concerned with Model 1's performance as a standalone model as we are with its effect on the later models. Thus, even though it is questionable whether variational Bayes improves Model 1 itself, we conclude that adding variational Bayes is good because it improves the performance of GIZA++ as a whole.

Since the main purpose of adding variational Bayes is to control overfitting, it is worth noting that overfitting is drastically reduced when variational Bayes is used. Figure 5 shows the test perplexity after GIZA++ has been run for twenty-five iterations of Model 1: without variational Bayes, the test perplexity increases dramatically, but it remains stable when variational Bayes is used. The

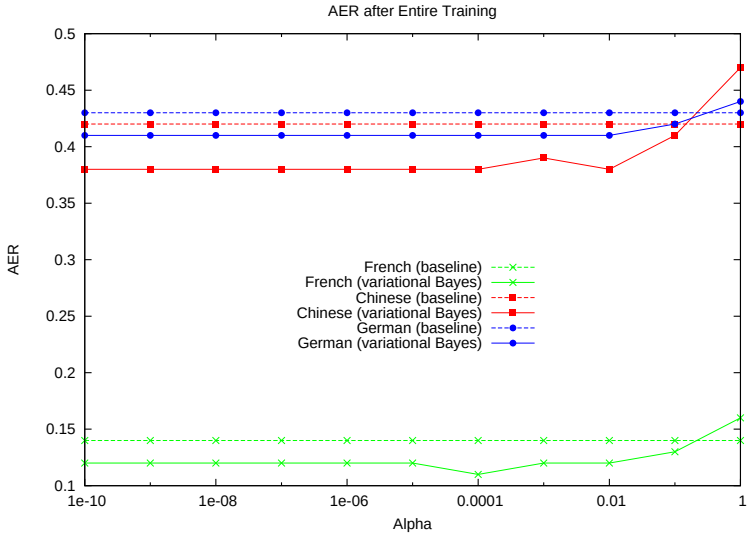

Figure 4: Determining the best value of α for the translation probabilities. Training data is 10000 sentence pairs from each language pair. Variational Bayes is used for Model 1 only. This table shows the AER for different values of α after training is complete (five iterations each of Models 1, HMM, 3, and 4).

other language pairs show similar patterns.

After choosing 0 as the best value of α for the translation probabilities, we reran the test above (five iterations each of Models 1, HMM, 3, and 4, with variational Bayes turned on for Model 1) on different amounts of data. We found that the results for larger data sizes were comparable to the results for ten thousand sentence pairs, both with and without variational Bayes, and both in terms of perplexity (Figure 6) and AER (Figure 7). Thus, for most of the rest of our tests, we only train GIZA++ on ten thousand sentence pairs, assuming the results to be generalizable to more realistic amounts of data.

We then examined the effects of variational Bayes on the later models, starting with Model 2. Model 2 uses alignment probabilities in addition to the translation probabilities of Model 1; thus, we ran a number of tests to determine the best value of α for the alignment probabilities. Using the same ten thousand sentence pairs for training as before, we ran GIZA++ for five iterations each of Models 1, 2, HMM, 3, and 4. Because the alignment probabilities interact with the translation probabilities, we ran two versions of this experiment: one with variational Bayes turned off for Model 2's translation probabilities, and one with it turned on. Figures 8 through 12 show the results of the first experiment; the results for the second were similar.

After training just Models 1 and 2, we find that values of α as high as 1 improve both the test perplexity and AER, but lower values of α result in higher test perplexity and AER than the baseline. As Figures 8 and 9 show, the test perplexity is higher than the baseline for most values of α, but as α approaches 1, the test perplexity drops dramatically, and is lower than the baseline for $\alpha = 1$.

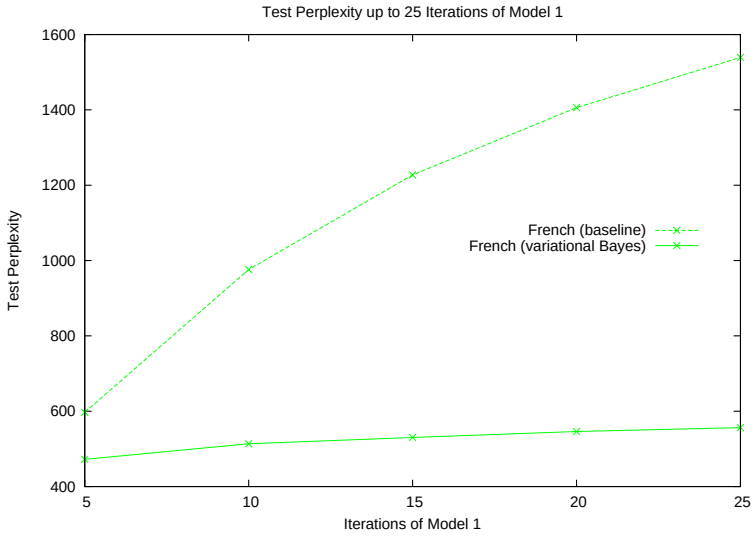

Figure 5: Effect of variational Bayes on overfitting for Model 1. Training data is 10000 sentence pairs. This table contrasts the test perplexities of Model 1 with variational Bayes and Model 1 without variational Bayes after different amounts of training for Model 1. Variational Bayes successfully controls overfitting.

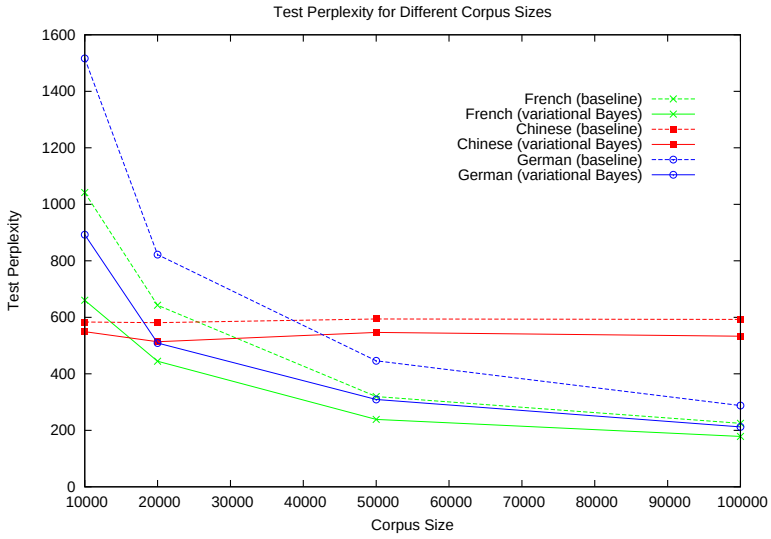

Figure 6: Performance of GIZA++ on different amounts of test data. Variational Bayes is used for Model 1 only. Table shows test perplexity after all the training has completed (five iterations each of Models 1, HMM, 3, and 4).

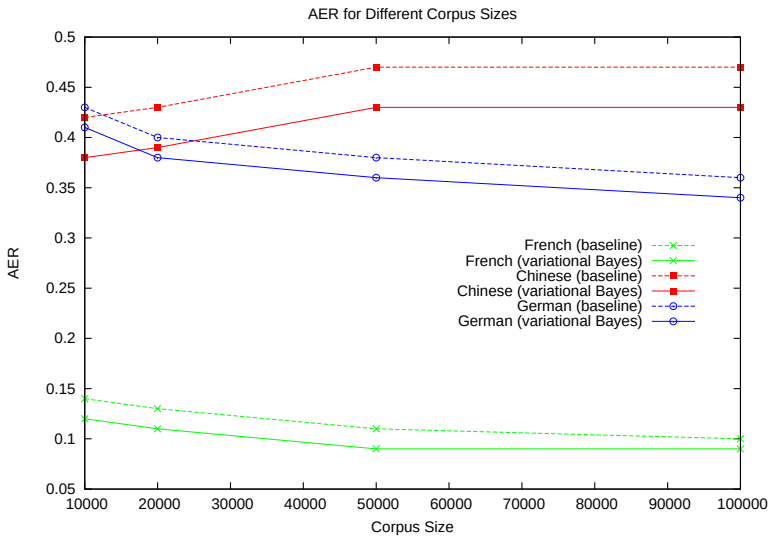

Figure 7: Performance of GIZA++ on different amounts of test data. Variational Bayes is used for Model 1 only. Table shows AER after all the training has completed (five iterations each of Models 1, HMM, 3, and 4).

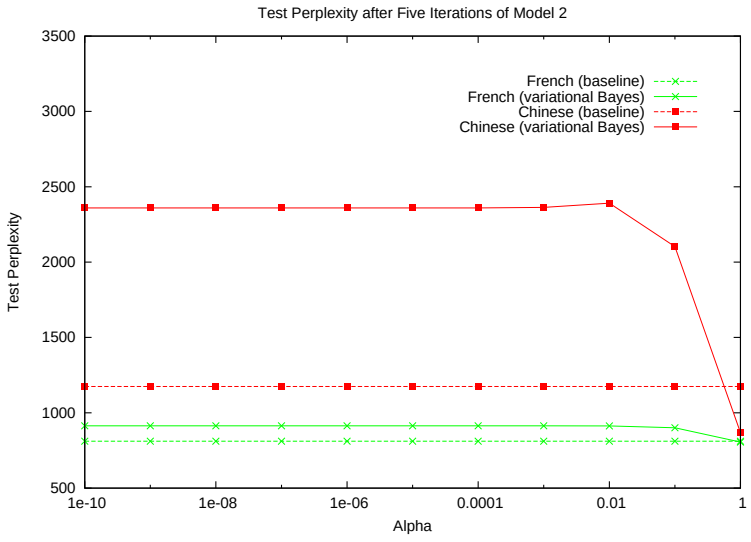

Figure 8: Results of varying α for Model 2 alignment probabilities. Training data is 10000 sentence pairs from each language pair. This table shows the test perplexity for French and Chinese after five iterations each of Model 1 and Model 2.

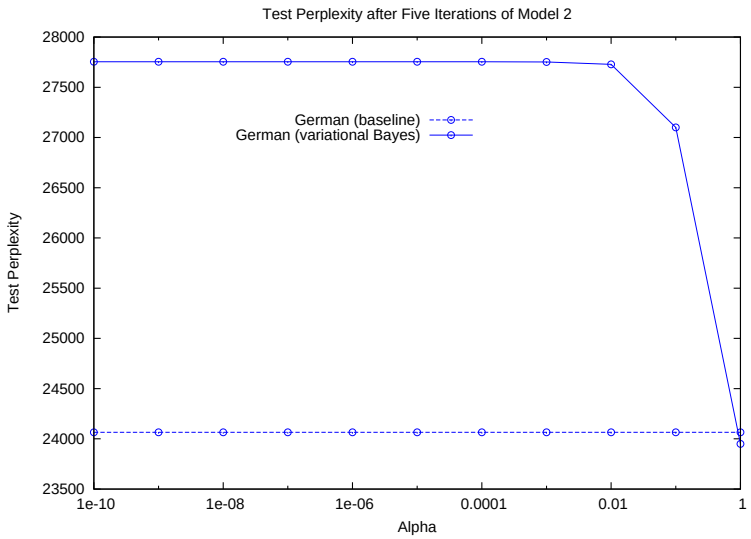

Figure 9: Results of varying α for Model 2 alignment probabilities. Training data is 10000 sentence pairs from each language pair. This table shows the test perplexity for German after five iterations each of Model 1 and Model 2.

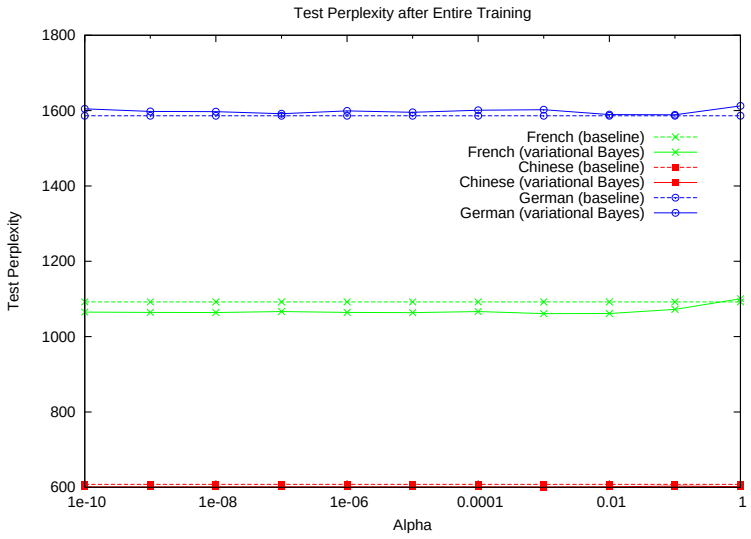

Figure 10: Results of varying α for Model 2 alignment probabilities. Training data is 10000 sentence pairs from each language pair. This table shows the test perplexity after training is complete (five iterations each of Models 1, 2, HMM, 3, and 4).

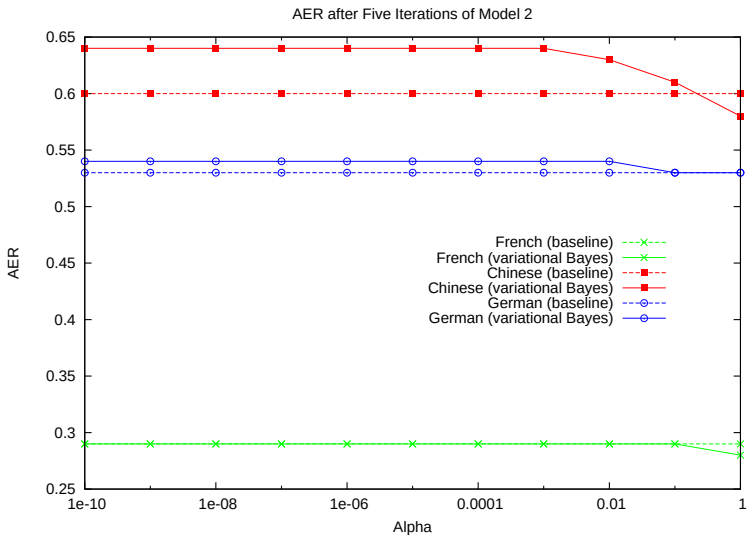

Figure 11: Results of varying α for Model 2 alignment probabilities. Training data is 10000 sentence pairs from each language pair. This table shows the AER after five iterations each of Model 1 and Model 2.

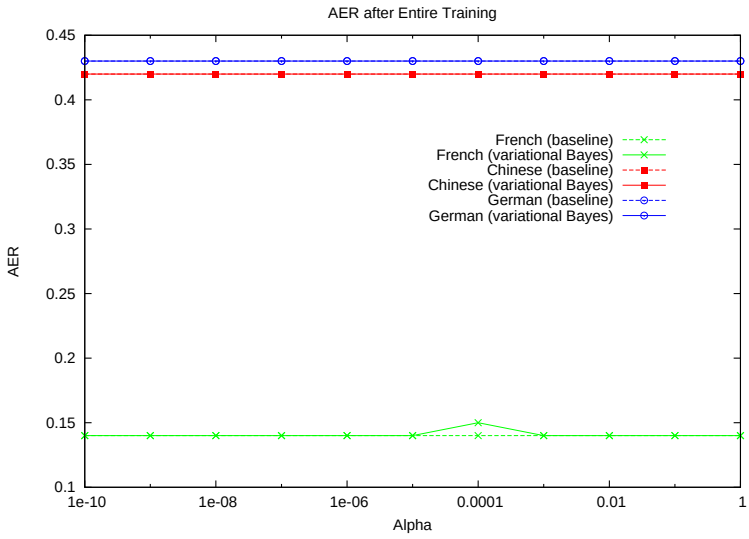

Figure 12: Results of varying α for Model 2 alignment probabilities. Training data is 10000 sentence pairs from each language pair. This table shows the AER after training is complete (five iterations each of Models 1, 2, HMM, 3, and 4).

	French		Chinese		German	
	TP	AER	TP	AER	TP	AER
Baseline System with Model 2	1092	0.14	607	0.42	1586	0.43
Baseline System without Model 2	1041	0.14	583	0.42	1516	0.43

Table 2: Ineffectiveness of Model 2

	French		Chinese		German	
	TP	AER	TP	AER	TP	AER
Baseline System	1092	0.14	607	0.42	1586	0.43
VB for Model 2 Alignment Tables	1061	0.14	601	0.42	1589	0.43
VB for Model 2 Translation Tables	1057	0.14	633	0.42	1466	0.42
VB for Both Model 2 Tables	1035	0.13	621	0.41	1473	0.42

Table 3: Effect of Variational Bayes on Alignment Tables

Figure 11 shows a similar result for AER: for lower values of α, the AER stays above or equal to the baseline, but as α approaches 1, the AER drops to or below the baseline.

But the opposite is true after all of the models have been trained. For lower values of α, test perplexity (Figure 10) stays very close to the baseline; the AER (Figure 12) also stays almost identical to the baseline. However, for higher values of α, test perplexity rises slightly for French and German. Thus we choose $\alpha = 0.01$, which gives slightly better results for the earlier models while not compromising the performance of the overall system.

More importantly, these tests reveal that Model 2 is not a good model. In both tests, the performance of the system suffers from its inclusion. Table 2 shows that the baseline system is not improved by adding Model 2; on the contrary, with the addition of Model 2 the test perplexity goes up for every model. Furthermore, in every test (with or without variational Bayes) that we ran, the test perplexity spiked dramatically upward after training Model 2, only to drop again after training the later models. For these reasons, we exclude Model 2 from the rest of our tests.

These tests also show that using variational Bayes for the alignment probabilities does not have a very large effect on the overall system. Table 3 compares the results of training Model 2 with and without variational Bayes for the alignment tables (α is set to 0.01 here). Adding variational Bayes for the alignment tables decreases the test perplexity for French, but increases it for German; AER does not change in most cases. Thus, in the rest of our tests, we do not use variational Bayes for the alignment probabilities, as its effect the on overall system is not significant enough to warrant its inclusion. We expect that variational Bayes for the alignment probabilities would have even less of an impact on the later models, where distortion probabilities are used instead of alignment probabilities, and the additional complexity of the models decreases the relative importance of alignment and distortion.

We then tested whether variational Bayes should be used for the later models. In all of these experiments, we ran Models 1, HMM, 3, and 4 for five iterations each, training on the same ten thousand sentence pairs that we used in the previous experiments. In Table 4, we show the performance of the system when no variational Bayes is used, when it is used for each of the four models individually, and when it is used for all four models simultaneously. We saw the most overall im-

	French		Chinese		German	
	TP	AER	TP	AER	TP	AER
Baseline	1041	0.14	583	0.42	1516	0.43
M1 Only	680	0.12	537	0.39	947	0.41
HMM Only	1044	0.14	644	0.42	1441	0.42
M3 Only	992	0.14	571	0.42	1472	0.43
M4 Only	1150	0.14	616	0.42	1670	0.43
All Models	491	0.19	568	0.44	417	0.45

Table 4: Effect of Variational Bayes on Later Models

	French		Chinese		German	
	TP	AER	TP	AER	TP	AER
Baseline	1041	0.14	583	0.42	1516	0.43
Baseline without M3	949	0.14	522	0.43	1308	0.40
M1 Only	680	0.12	537	0.39	947	0.41
M1 Only without M3	603	0.12	498	0.40	782	0.39
All Models	491	0.19	568	0.44	417	0.45
All Models without M3	474	0.18	549	0.44	406	0.44

Table 5: Ineffectiveness of Model 3

provement when variational Bayes was used only for Model 1; using variational Bayes for all four models simultaneously caused the most improvement to the test perplexity, but at the cost of the AER.

In all of these experiments, both the test perplexity and the AER increased after Model 3 was trained, only to decrease again after training Model 4. Thus, we tested what happens when Model 3 is removed from the baseline and the two best settings of the system, and found that the system performs better without Model 3. As can be seen in Table 5, removing Model 3 lowers the test perplexity for each model and for each language, and improves or does not alter the AER for almost every model and language pair. As a result, we exclude Model 3 from our MT experiments.

For the MT experiments, we ran GIZA++ through Moses, training Model 1, the HMM, and Model 4 on 100000 sentence pairs from each language pair. We ran three experiments, one with variational Bayes turned on for all models, one with variational Bayes turned on for Model 1 only, and one (the baseline) with variational Bayes turned off for all models. When variational Bayes was turned on, we ran GIZA++ for five iterations per model as in our earlier tests, but when variational Bayes was turned off, we ran GIZA++ for only four iterations per model, having determined that this was the optimal number of iterations for baseline system. Variational Bayes was used for the translation probabilities only, with α set to 0.

As can be seen in Table 6, using variational Bayes causes an increase in BLEU score for all three language pairs. For French, the best results were achieved when variational Bayes was used for Model 1 only; for Chinese and German, on the other hand, using variational Bayes for all models caused the most improvements. For French, the BLEU score increased by 0.20; for German, it increased by 0.82; for Chinese, it increased by 1.05. Overall, variational Bayes seems to have the

	BLEU Score		
	French	Chinese	German
Baseline	26.34	21.03	21.14
M1 Only	26.54	21.58	21.73
All Models	26.46	22.08	21.96

Table 6: BLEU Scores

greatest impact for the language pairs that are most difficult to align and translate to begin with.

5 Running GIZA++

Our version of GIZA++ works almost identically to the original program. When running GIZA++ without variational Bayes, our version can be used just like the original code, as all of our variational Bayes parameters are turned off by default. When running GIZA++ with variational Bayes, there are a number of parameters that can be set that determine which Models should use variational Bayes and for what tables, as well as what the values of α should be for each Model. The new parameters are as follows:

- **model1tvb**, **model2tvb**, **modelhmmtvb**, **model2to3tvb**, **model3tvb**, **model4tvb**, **model5tvb** - these parameters control whether variational Bayes is used for the translation probabilities $t(f|e)$. There is one parameter for each IBM Model and one for the HMM. The **model2to3tvb** parameter determines whether variational Bayes is used when switching from Model 2 to Model 3; this process only occurs if the HMM is not used for training. These parameters default to 0 (variational Bayes off); when they are set to 1, variational Bayes is turned on.

- **model1talpha**, **model2talpha**, **modelhmmtalpha**, **model2to3talpha**, **model3talpha**, **model4talpha**, **model5talpha** - these parameters determine the value of α used when training the translation tables for the different models with variational Bayes. α parameters default to 0, as our results show that the translation table performs best with as low a value of α as possible. When variational Bayes is turned off, the value of these parameters is ignored.

- **model1tvbnorm**, **model2tvbnorm**, **modelhmmtvbnorm**, **model2to3tvbnorm**, **model3tvbnorm**, **model4tvbnorm**, **model5tvbnorm** - these parameters determine whether an extra round of normalization is performed after variational Bayes is used. This extra normalization ensures that the probabilities will add up to one, which is useful for calculating the perplexities, as discussed above. However, it does detract slightly from the effect of variational Bayes. These parameters default to 0 (extra normalization off); they can be set to 1 to turn extra normalization on. Again, these parameters are ignored when variational Bayes is not being used.

- **model2avb**, **modelhmmavb** - these parameters determine whether variational Bayes is used for the alignment table, which stores alignment probabilities $a(i|j, l, m)$. We have only added variational Bayes to the alignment tables for Model 2 and the HMM, since we found that

using variational Bayes for the alignment tables had relatively little effect on the program's performance. Like the translation table variational Bayes parameters, these default to 0 and can be set to 1 to turn on variational Bayes.

- **model2aalpha**, **modelhmmaalpha** - these parameters determine the value of α used for the alignment tables when running Model 2 and the HMM with variational Bayes. Like the translation table alpha parameters, these default to 0, though higher values of α can be more effective when using variational Bayes for alignment probabilities.

Thus, running GIZA++ with variational Bayes simply requires setting a few extra parameters.

It is easy to incorporate our version of GIZA++ into the statistical machine translation system Moses. Moses takes the location of the GIZA++ binary as a parameter, making it simple to substitute our version of GIZA++ for the original. Moses also uses a parameter called "–giza-option" to specify the parameters that should be supplied to GIZA++; these are passed directly to GIZA++ and thus GIZA++ can easily be run inside Moses with our variational Bayes parameters.

6 Conclusion

We find that applying Variational Bayes with a Dirichlet prior to the translation models implemented in GIZA++ improves alignments, both in terms of AER and the BLEU score of an end-to-end translation system. Variational Bayes is particularly beneficial for IBM Model 1, and applying VB to Model 1 alone tends to improve the performance of later models in the training sequence.

To understand why VB is particularly important for Model 1, consider again the test case described in Table 1. Intuitively, we as humans know that the correct translation of f_2 is probably e_2, and that e_1 only translates to f_1, even if we have only seen one instance of each sentence pair. This is because we recognize that words have *fertility*; given a sentence pair where each sentence contains two words, we expect each word to have a single translation.

Model 1 is particularly sensitive to overfitting because it does not use information such as fertility; the later Models, which incorporate additional information, are much less susceptible to overfitting. This helps explain why variational Bayes is particularly important for Model 1, and why it has less of an effect on the later Models. Using variational Bayes for Model 1 improves the performance of the later Models, confirming Moore's suggestion that it is important to keep Model 1 from overfitting so as not to bias the later Models towards incorrect solutions.

Future research on this topic might involve adding Moore's fixes for Model 1 to GIZA++ and comparing them to variational Bayes. We would also want to run more tests with actual machine translation systems, and perhaps to optimize α and the number of iterations for each Model based on BLEU score rather than perplexity and AER.

References

Baum, Leonard E. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In Oved Shisha, editor, *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8. Academic Press, University of California, Los Angeles.

Beal, Matthew J. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, University College London.

Blunsom, Phil, Trevor Cohn, and Miles Osborne. 2008. Bayesian synchronous grammar induction. In *Neural Information Processing Systems (NIPS)*.

Brown, Peter F., 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.

Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL-05*, pages 263–270. Ann Arbor, MI.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–21.

DeNero, John, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 314–323. Honolulu, Hawaii.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of NAACL-04*, pages 273–280.

Johnson, Mark. 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. Association for Computational Linguistics, Prague, Czech Republic.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL, Demonstration Session*, pages 177–180.

Moore, Robert C. 2004. Improving ibm word alignment model 1. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 518–525. Barcelona, Spain.

Och, Franz Josef and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL-00*, pages 440–447. Hong Kong.

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

Vogel, Stephan, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *COLING-96*, pages 836–841.