Alignment by Agreement

Percy Liang
UC Berkeley
Berkeley, CA 94720
pliang@cs.berkeley.edu

Ben Taskar
UC Berkeley
Berkeley, CA 94720
taskar@cs.berkeley.edu

Dan Klein
UC Berkeley
Berkeley, CA 94720
klein@cs.berkeley.edu

Abstract

We present an unsupervised approach to symmetric word alignment in which two simple asymmetric models are trained jointly to maximize a combination of data likelihood and agreement between the models. Compared to the standard practice of intersecting predictions of independently-trained models, joint training provides a 32% reduction in AER. Moreover, a simple and efficient pair of HMM aligners provides a 29% reduction in AER over symmetrized IBM model 4 predictions.

1 Introduction

Word alignment is an important component of a complete statistical machine translation pipeline (Koehn et al., 2003). The classic approaches to unsupervised word alignment are based on IBM models 1–5 (Brown et al., 1994) and the HMM model (Ney and Vogel, 1996) (see Och and Ney (2003) for a systematic comparison). One can classify these six models into two groups: sequence-based models (models 1, 2, and HMM) and fertility-based models (models 3, 4, and 5).[1] Whereas the sequence-based models are tractable and easily implemented, the more accurate fertility-based models are intractable and thus require approximation methods which are difficult to implement. As a result, many practitioners use the complex GIZA++ software package (Och and Ney, 2003) as a black box, selecting model 4 as a good compromise between alignment quality and efficiency.

Even though the fertility-based models are more accurate, there are several reasons to consider avenues for improvement based on the simpler and faster sequence-based models. First, even with the highly optimized implementations in GIZA++, models 3 and above are still very slow to train. Second, we seem to have hit a point of diminishing returns with extensions to the fertility-based models. For example, gains from the new model 6 of Och and Ney (2003) are modest. When models are too complex to reimplement, the barrier to improvement is raised even higher. Finally, the fertility-based models are asymmetric, and symmetrization is commonly employed to improve alignment quality by intersecting alignments induced in each translation direction. It is therefore natural to explore models which are designed from the start with symmetry in mind.

In this paper, we introduce a new method for word alignment that addresses the three issues above. Our development is motivated by the observation that intersecting the predictions of two directional models outperforms each model alone. Viewing intersection as a way of finding predictions that both models agree on, we take the agreement idea one step further. The central idea of our approach is to not only make the predictions of the models agree at test time, but also encourage agreement during training. We define an intuitive objective function which incor-

[1] IBM models 1 and 2 are considered sequence-based models because they are special cases of HMMs with transitions that do not depend on previous states.

porates both data likelihood and a measure of agreement between models. Then we derive an EM-like algorithm to maximize this objective function. Because the E-step is intractable in our case, we use a heuristic approximation which nonetheless works well in practice.

By jointly training two simple HMM models, we obtain 4.9% AER on the standard English-French Hansards task. To our knowledge, this is the lowest published unsupervised AER result, and it is competitive with supervised approaches. Furthermore, our approach is very practical: it is no harder to implement than a standard HMM model, and joint training is no slower than the standard training of two HMM models. Finally, we show that word alignments from our system can be used in a phrase-based translation system to modestly improve BLEU score.

2 Alignment models: IBM 1, 2 and HMM

We briefly review the sequence-based word alignment models (Brown et al., 1994; Och and Ney, 2003) and describe some of the choices in our implementation. All three models are generative models of the form $p(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a}} p(\mathbf{a}, \mathbf{f} \mid \mathbf{e})$, where $\mathbf{e} = (e_1, \ldots, e_I)$ is the English sentence, $\mathbf{f} = (f_1, \ldots, f_J)$ is the French sentence, and $\mathbf{a} = (a_1, \ldots, a_J)$ is the (asymmetric) alignment which specifies the position of an English word aligned to each French word. All three models factor in the following way:

$$p(\mathbf{a}, \mathbf{f} \mid \mathbf{e}) = \prod_{j=1}^{J} p_{\mathrm{d}}(a_j \mid a_{j_-}, j) p_{\mathrm{t}}(f_j \mid e_{a_j}), \quad (1)$$

where j_- is the position of the last non-null-aligned French word before position j.[2]

The translation parameters $p_{\mathrm{t}}(f_j \mid e_{a_j})$ are parameterized by an (unsmoothed) lookup table that stores the appropriate local conditional probability distributions. The distortion parameters $p_{\mathrm{d}}(a_j = i' \mid a_{j_-} = i)$ depend on the particular model (we write $a_j = 0$ to denote the event that the j-th French word is null-aligned):

$$\begin{aligned} p_{\mathrm{d}}(a_j = 0 \mid a_{j_-} = i) &= p_0 \\ p_{\mathrm{d}}(a_j = i' \neq 0 \mid a_{j_-} = i) &\propto \\ (1 - p_0) \cdot & \begin{cases} 1 & \text{(IBM 1)} \\ c(i' - \lfloor \frac{jI}{J} \rfloor) & \text{(IBM 2)} \\ c(i' - i) & \text{(HMM)}, \end{cases} \end{aligned}$$

where p_0 is the null-word probability and $c(\cdot)$ contains the distortion parameters for each offset argument. We set the null-word probability $p_0 = \frac{1}{I+1}$ depending on the length of the English sentence, which we found to be more effective than using a constant p_0.

In model 1, the distortion $p_{\mathrm{d}}(\cdot \mid \cdot)$ specifies a uniform distribution over English positions. In model 2, $p_{\mathrm{d}}(\cdot \mid \cdot)$ is still independent of a_{j_-}, but it can now depend on j and i' through $c(\cdot)$. In the HMM model, there is a dependence on $a_{j_-} = i$, but only through $c(i - i')$.

We parameterize the distortion $c(\cdot)$ using a multinomial distribution over 11 offset buckets $c(\leq -5), c(-4), \ldots, c(4), c(\geq 5)$.[3] We use three sets of distortion parameters, one for transitioning into the first state, one for transitioning out of the last state, and one for all other transitions. This works better than using a single set of parameters or ignoring the transitions at the two ends.

3 Training by agreement

To motivate our joint training approach, we first consider the standard practice of intersecting alignments. While the English and French sentences play a symmetric role in the word alignment task, sequence-based models are asymmetric: they are generative models of the form $p(\mathbf{f} \mid \mathbf{e})$ (E→F), or $p(\mathbf{e} \mid \mathbf{f})$ (F→E) by reversing the roles of source and target. In general, intersecting the alignment predictions of two independently-trained directional models reduces AER, e.g., from 11% to 7% for HMM models (Table 2). This suggests that two models make different types of errors that can be eliminated upon intersection. Figure 1 (top) shows a common type of error that intersection can partly remedy. In

[2] The dependence on a_{j_-} can in fact be implemented as a first-order HMM (see Och and Ney (2003)).

[3] For each sentence, the probability mass of each of the two end buckets $c(\leq -5)$ or $c(\geq 5)$ is uniformly divided among those valid offsets.

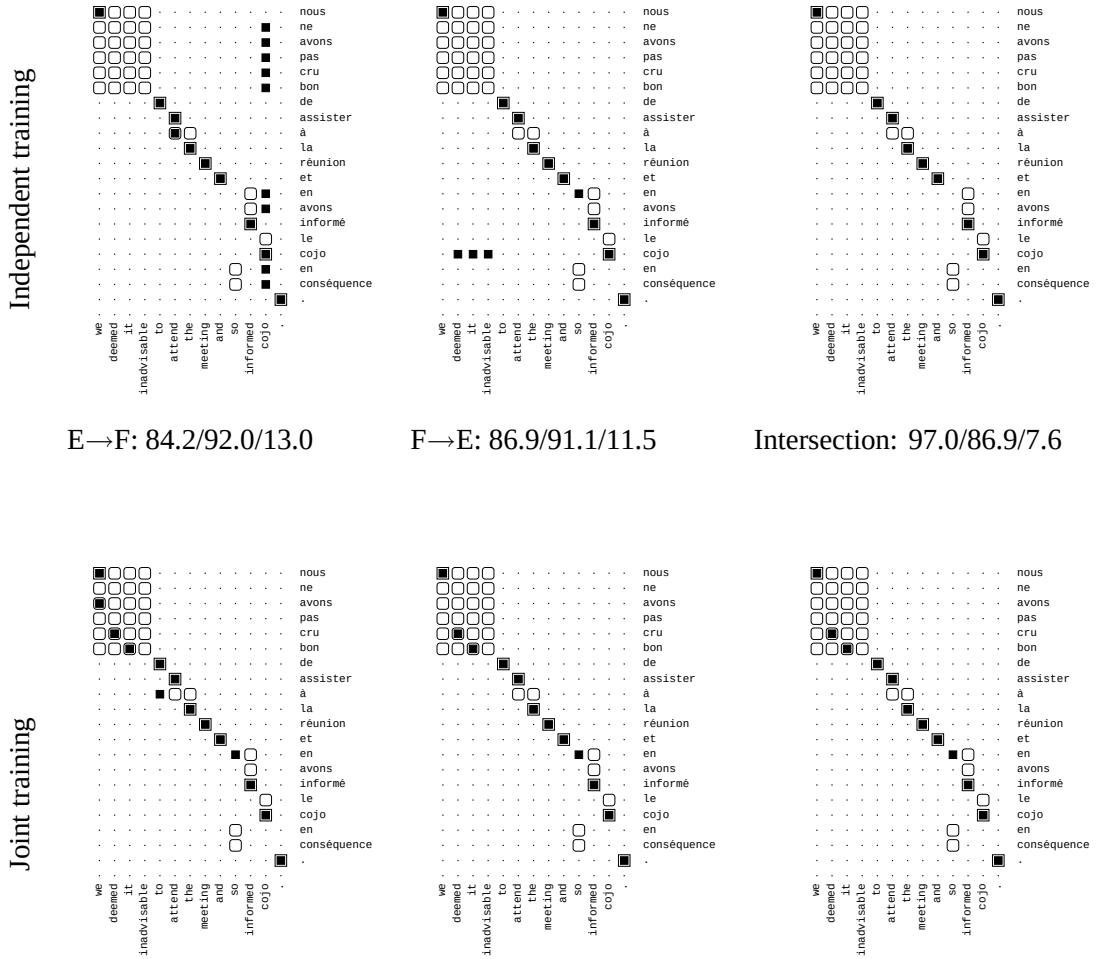

Figure 1: An example of the Viterbi output of a pair of independently trained HMMs (top) and a pair of jointly trained HMMs (bottom), both trained on 1.1 million sentences. Rounded boxes denote possible alignments, square boxes are sure alignments, and solid boxes are model predictions. For each model, the overall Precision/Recall/AER on the development set is given. See Section 4 for details.

this example, *COJO* is a rare word that becomes a garbage collector (Moore, 2004) for the models in both directions. Intersection eliminates the spurious alignments, but at the expense of recall.

Intersection after training produces alignments that both models agree on. The joint training procedure we describe below builds on this idea by encouraging the models to agree during training. Consider the output of the jointly trained HMMs in Figure 1 (bottom). The garbage-collecting rare word is no longer a problem. Not only are the individual E→F and F→E jointly-trained models better than their independently-trained counterparts, the jointly-trained intersected model also provides a significant overall gain over the independently-trained intersected model. We maintain both high precision and recall.

Before we introduce the objective function for joint training, we will write the two directional models in a symmetric way so that they share the same

alignment spaces. We first replace the asymmetric alignments $\mathbf{a}$ with a set of indicator variables for each potential alignment edge (i, j): $\mathbf{z} = \{z_{ij} \in \{0, 1\} : 1 \leq i \leq I, 1 \leq j \leq J\}$. Each $\mathbf{z}$ can be thought of as an element in the set of *generalized alignments*, where any subset of word pairs may be aligned (Och and Ney, 2003). Sequence-based models $p(\mathbf{a} \mid \mathbf{e}, \mathbf{f})$ induce a distribution over $p(\mathbf{z} \mid \mathbf{e}, \mathbf{f})$ by letting $p(\mathbf{z} \mid \mathbf{e}, \mathbf{f}) = 0$ for any $\mathbf{z}$ that does not correspond to any $\mathbf{a}$ (i.e., if $\mathbf{z}$ contains many-to-one alignments).

We also introduce the more compact notation $\mathbf{x} = (\mathbf{e}, \mathbf{f})$ to denote an input sentence pair. We put arbitrary distributions $p(\mathbf{e})$ and $p(\mathbf{f})$ to remove the conditioning, noting that this has no effect on the optimization problem in the next section. We can now think of the two directional sequence-based models as each inducing a distribution over the same space of sentence pairs and alignments $(\mathbf{x}, \mathbf{z})$:

$$p_1(\mathbf{x}, \mathbf{z}; \theta_1) = p(\mathbf{e})p(\mathbf{a}, \mathbf{f} \mid \mathbf{e}; \theta_1)$$

$$p_2(\mathbf{x}, \mathbf{z}; \theta_2) = p(\mathbf{f})p(\mathbf{a}, \mathbf{e} \mid \mathbf{f}; \theta_2).$$

3.1 A joint objective

In the next two sections, we describe how to jointly train the two models using an EM-like algorithm. We emphasize that this technique is quite general and can be applied in many different situations where we want to couple two tractable models over input $\mathbf{x}$ and output $\mathbf{z}$.

To train two models $p_1(\mathbf{x}, \mathbf{z}; \theta_1)$ and $p_2(\mathbf{x}, \mathbf{z}; \theta_2)$ independently, we maximize the data likelihood $\prod_{\mathbf{x}} p_k(\mathbf{x}; \theta_k) = \prod_{\mathbf{x}} \sum_{\mathbf{z}} p_k(\mathbf{x}, \mathbf{z}; \theta_k)$ of each model separately, $k \in \{1, 2\}$:

$$\max_{\theta_1, \theta_2} \sum_{\mathbf{x}} \left[\log p_1(\mathbf{x}; \theta_1) + \log p_2(\mathbf{x}; \theta_2)\right]. \quad (2)$$

Above, the summation over $\mathbf{x}$ enumerates the sentence pairs in the training data.

There are many possible ways to quantify agreement between two models. We chose a particularly simple and mathematically convenient measure — the probability that the alignments produced by the two models agree on an example $\mathbf{x}$:

$$\sum_{\mathbf{z}} p_1(\mathbf{z} \mid \mathbf{x}; \theta_1) p_2(\mathbf{z} \mid \mathbf{x}; \theta_2).$$

We add the (log) probability of agreement to the standard log-likelihood objective to couple the two models:

$$\max_{\theta_1, \theta_2} \sum_{\mathbf{x}} [\log p_1(\mathbf{x}; \theta_1) + \log p_2(\mathbf{x}; \theta_2) + \log \sum_{\mathbf{z}} p_1(\mathbf{z} \mid \mathbf{x}; \theta_1) p_2(\mathbf{z} \mid \mathbf{x}; \theta_2)]. \quad (3)$$

3.2 Optimization via EM

We first review the EM algorithm for optimizing a single model, which consists of iterating the following two steps:

$$\begin{aligned} \text{E}: &\quad q(\mathbf{z}; \mathbf{x}) := p(\mathbf{z} \mid \mathbf{x}; \theta), \\ \text{M}: &\quad \theta' := \operatorname*{argmax}_{\theta} \sum_{\mathbf{x}, \mathbf{z}} q(\mathbf{z}; \mathbf{x}) \log p(\mathbf{x}, \mathbf{z}; \theta). \end{aligned}$$

In the E-step, we compute the posterior distribution of the alignments $q(\mathbf{z}; \mathbf{x})$ given the sentence pair $\mathbf{x}$ and current parameters θ. In the M-step, we use expected counts with respect to $q(\mathbf{z}; \mathbf{x})$ in the maximum likelihood update $\theta := \theta'$.

To optimize the objective in Equation 3, we can derive a similar and simple procedure. See the appendix for the derivation.

$$\begin{aligned} \text{E: } & q(\mathbf{z}; \mathbf{x}) := \tfrac{1}{Z_{\mathbf{x}}} p_1(\mathbf{z} \mid \mathbf{x}; \theta_1) p_2(\mathbf{z} \mid \mathbf{x}; \theta_2), \\ \text{M: } & \theta' = \operatorname*{argmax}_{\theta} \sum_{\mathbf{x}, \mathbf{z}} q(\mathbf{z}; \mathbf{x}) \log p_1(\mathbf{x}, \mathbf{z}; \theta_1) \\ & \qquad + \sum_{\mathbf{x}, \mathbf{z}} q(\mathbf{z}; \mathbf{x}) \log p_2(\mathbf{x}, \mathbf{z}; \theta_2), \end{aligned}$$

where $Z_{\mathbf{x}}$ is a normalization constant. The M-step decouples neatly into two independent optimization problems, which lead to single model updates using the expected counts from $q(\mathbf{z}; \mathbf{x})$. To compute $Z_{\mathbf{x}}$ in the E-step, we must sum the product of two model posteriors over the set of possible $\mathbf{z}$s with nonzero probability under both models. In general, if both posterior distributions over the latent variables $\mathbf{z}$ decompose in the same tractable manner, as in the context-free grammar induction work of Klein and Manning (2004), the summation could be carried out efficiently, for example using dynamic programming. In our case, we would have to sum over the set of alignments where each word in English is aligned to at most one word in French and each word in French is aligned to at most one

word in English. Unfortunately, for even very simple models such as IBM 1 or 2, computing the normalization constant over this set of alignments is a $\#P$-complete problem, by a reduction from counting matchings in a bipartite graph (Valiant, 1979). We could perhaps attempt to compute q using a variety of approximate probabilistic inference techniques, for example, sampling or variational methods. With efficiency as our main concern, we opted instead for a simple heuristic procedure by letting q be a product of marginals:

$$q(\mathbf{z};\mathbf{x}) := \prod_{i,j} p_1(z_{ij} \mid \mathbf{x};\theta_1)p_2(z_{ij} \mid \mathbf{x};\theta_2),$$

where each $p_k(z_{ij} \mid \mathbf{x};\theta_k)$ is the posterior marginal probability of the (i,j) edge being present (or absent) in the alignment according to each model, which can be computed separately and efficiently.

Now the new E-step only requires simple marginal computations under each of the models. This procedure is very intuitive: edges on which the models disagree are discounted in the E-step because the product of the marginals $p_1(z_{ij} \mid \mathbf{x};\theta_1)p_2(z_{ij} \mid \mathbf{x};\theta_2)$ is small. Note that in general, this new procedure is not guaranteed to increase our joint objective. Nonetheless, our experimental results show that it provides an effective method of achieving model agreement and leads to significant accuracy gains over independent training.

3.3 Prediction

Once we have trained two models, either jointly or independently, we must decide how to combine those two models to predict alignments for new sentences.

First, let us step back to the case of one model. Typically, the Viterbi alignment $\operatorname{argmax}_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{x})$ is used. An alternative is to use posterior decoding, where we keep an edge (i,j) if the marginal edge posterior $p(z_{ij} \mid \mathbf{x})$ exceeds some threshold $0 < \delta < 1$. In symbols, $\mathbf{z} = \{z_{ij} = 1 : p(z_{ij} = 1 \mid \mathbf{x}) \geq \delta\}$.[4]

Posterior decoding has several attractive advantages over Viterbi decoding. Varying the threshold δ gives a natural way to tradeoff precision and recall. In fact, these posteriors could be used more directly in extracting phrases for phrase-based translation. Also, when we want to combine two models for prediction, finding the Viterbi alignment $\operatorname{argmax}_{\mathbf{z}} p_1(\mathbf{z} \mid \mathbf{x})p_2(\mathbf{z} \mid \mathbf{x})$ is intractable for HMM models (by a reduction from quadratic assignment), and a hard intersection $\operatorname{argmax}_{\mathbf{z}_1} p_1(\mathbf{z}_1 \mid \mathbf{x}) \cap \operatorname{argmax}_{\mathbf{z}_2} p_2(\mathbf{z}_2 \mid \mathbf{x})$ might be too sparse. On the other hand, we can threshold the product of two edge posteriors quite easily: $\mathbf{z} = \{z_{ij} = 1 : p_1(z_{ij} = 1 \mid \mathbf{x})p_2(z_{ij} = 1 \mid \mathbf{x}) \geq \delta\}$.

We noticed a 5.8% relative reduction in AER (for our best model) by using posterior decoding with a validation-set optimized threshold δ instead of using hard intersection of Viterbi alignments.

4 Experiments

We tested our approach on the English-French Hansards data from the NAACL 2003 Shared Task, which includes a training set of 1.1 million sentences, a validation set of 37 sentences, and a test set of 447 sentences. The validation and test sentences have been hand-aligned (see Och and Ney (2003)) and are marked with both *sure* and *possible* alignments. Using these alignments, *alignment error rate* (AER) is calculated as:

$$\left(1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}\right) \times 100\%,$$

where A is a set of proposed edges, S is the sure gold edges, and P is the possible gold edges.

As a preprocessing step, we lowercased all words. Then we used the validation set and the first 100 sentences of the test set as our development set to tune our models. Lastly, we ran our models on the last 347 sentences of the test set to get final AER results.

4.1 Basic results

We trained models 1, 2, and HMM on the Hansards data. Following past work, we initialized the translation probabilities of model 1 uniformly over word pairs that occur together in some sentence pair. Models 2 and HMM were initialized with uniform distortion probabilities and model 1 translation probabilities. Each model was trained for 5 iterations, using the same training regimen as in Och and Ney (2003).

[4] See Matusov et al. (2004) for an alternative use of these marginals.

Model	Indep.	Joint	Reduction
10K sentences			
Model 1	27.4	23.6	13.8
Model 2	18.2	14.9	18.5
HMM	12.1	8.4	30.6
100K sentences			
Model 1	21.5	19.2	10.9
Model 2	13.1	10.2	21.7
HMM	8.0	5.3	33.1
1.1M sentences			
Model 1	20.0	16.5	17.5
Model 2	11.4	9.2	18.8
HMM	6.6	5.2	21.5

Table 1: Comparison of AER between independent and joint training across different size training sets and different models, evaluated on the development set. The last column shows the relative reduction in AER.

Table 1 shows a summary of the performance of independently and jointly trained models under various training conditions. Quite remarkably, for all training data sizes and all of the models, we see an appreciable reduction in AER, especially on the HMM models. We speculate that since the HMM model provides a richer family of distributions over alignments than either models 1 or 2, we can learn to synchronize the predictions of the two models, whereas models 1 and 2 have a much more limited capacity to synchronize.

Table 2 shows the HMM models compared to model 4 alignments produced by GIZA++ on the test set. Our jointly trained model clearly outperforms not only the standard HMM but also the more complex IBM 4 model. For these results, the threshold used for posterior decoding was tuned on the development set. "GIZA HMM" and "HMM, indep" are the same algorithm but differ in implementation details. The E→F and F→E models benefit a great deal by moving from independent to joint training, and the combined models show a smaller improvement.

Our best performing model differs from standard IBM word alignment models in two ways. First and most importantly, we use joint training instead of independent training, which gives us a huge boost. The second change, which is more minor and orthogonal, is using posterior decoding instead of Viterbi decoding, which also helps performance for model 2 and HMM, but not model 1. Table 3 quantifies the contribution of each of these two dimensions.

Model	E→F	F→E	Combined
GIZA HMM	11.5	11.5	7.0
GIZA Model 4	8.9	9.7	6.9
HMM, indep	11.2	11.5	7.2
HMM, joint	6.1	6.6	**4.9**

Table 2: Comparison of test set AER between various models trained on the full 1.1 million sentences.

Model	I+V	I+P	J+V	J+P
10K sentences				
Model 1	29.4	27.4	**22.7**	23.6
Model 2	20.1	18.2	16.5	**14.9**
HMM	15.2	12.1	8.9	**8.4**
100K sentences				
Model 1	22.9	21.5	**18.6**	19.2
Model 2	15.1	13.1	12.9	**10.2**
HMM	9.2	8.0	6.0	**5.3**
1.1M sentences				
Model 1	20.0	19.4	**16.5**	17.3
Model 2	12.7	11.4	11.6	**9.2**
HMM	7.6	6.6	5.7	**5.2**

Table 3: Contributions of using joint training versus independent training and posterior decoding (with the optimal threshold) instead of Viterbi decoding, evaluated on the development set.

Posterior decoding In our results, we have tuned our threshold to minimize AER. It turns out that AER is relatively insensitive to the threshold as Figure 2 shows. There is a large range from 0.2 to 0.5 where posterior decoding outperforms Viterbi decoding.

Initialization and convergence In addition to improving performance, joint training also enjoys certain robustness properties. Specialized initialization is absolutely crucial for an independently-trained

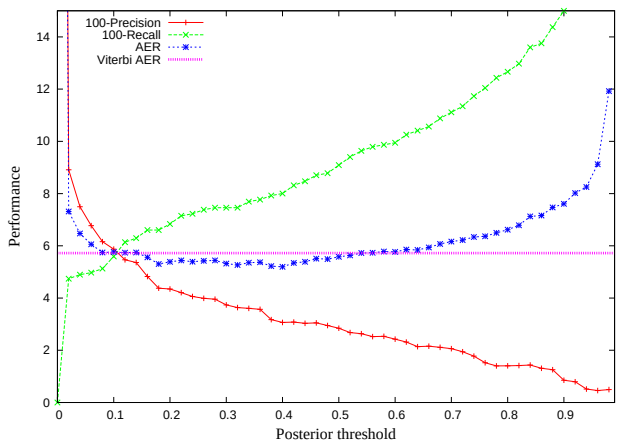

Figure 2: The precision, recall, and AER as the threshold is varied for posterior decoding in a jointly trained pair of HMMs.

HMM model. If we initialize the HMM model with uniform translation parameters, the HMM converges to a completely senseless local optimum with AER above 50%. Initializing the HMM with model 1 parameters alleviates this problem.

On the other hand, if we jointly train two HMMs starting from a uniform initialization, the HMMs converge to a surprisingly good solution. On the full training set, training two HMMs jointly from uniform initialization yields 5.7% AER, only slightly higher than 5.2% AER using model 1 initialization. We suspect that the agreement term of the objective forces the two HMMs to avoid many local optima that each one would have on its own, since these local optima correspond to posteriors over alignments that would be very unlikely to agree. We also observed that jointly trained HMMs converged very quickly—in 5 iterations—and did not exhibit overfitting with increased iterations.

Common errors The major source of remaining errors are recall errors that come from the shortcomings of the HMM model. The E→F model gives 0 probability to any many-to-one alignments and the F→E model gives 0 probability to any one-to-many alignments. By enforcing agreement, the two models are effectively restricted to one-to-one (or zero) alignments. Posterior decoding is in principle capable of proposing many-to-many alignments, but these alignments occur infrequently since the posteriors are generally sharply peaked around the Viterbi alignment. In some cases, however, we do get one-to-many alignments in both directions.

Another common type of errors are precision errors due to the models overly-aggressively preferring alignments that preserve monotonicity. Our HMM model only uses 11 distortion parameters, which means distortions are not sensitive to the lexical context of the sentences. For example, in one sentence, *le* is incorrectly aligned to *the* as a monotonic alignment following another pair of correctly aligned words, and then the monotonicity is broken immediately following *le–the*. Here, the model is insensitive to the fact that alignments following articles tend to be monotonic, but alignments preceding articles are less so.

Another phenomenon is the insertion of "stepping stone" alignments. Suppose two edges (i, j) and $(i{+}4, j{+}4)$ have a very high probability of being included in an alignment, but the words between them are not good translations of each other. If the intervening English words were null-aligned, we would have to pay a big distortion penalty for jumping 4 positions. On the other hand, if the edge $(i{+}2, j{+}2)$ were included, that penalty would be mitigated. The translation cost for forcing that edge is smaller than the distortion cost.

4.2 BLEU evaluation

To see whether our improvement in AER also improves BLEU score, we aligned 100K English-French sentences from the Europarl corpus and tested on 3000 sentences of length 5–15. Using GIZA++ model 4 alignments and Pharaoh (Koehn et al., 2003), we achieved a BLEU score of 0.3035. By using alignments from our jointly trained HMMs instead, we get a BLEU score of 0.3051. While this improvement is very modest, we are currently investigating alternative ways of interfacing with phrase table construction to make a larger impact on translation quality.

5 Related Work

Our approach is similar in spirit to co-training, where two classifiers, complementary by the virtue of having different views of the data, are trained jointly to encourage agreement (Blum and Mitchell, 1998; Collins and Singer, 1999). One key difference

in our work is that we rely exclusively on data likelihood to guide the two models in an unsupervised manner, rather than relying on an initial handful of labeled examples.

The idea of exploiting agreement between two latent variable models is not new; there has been substantial previous work on leveraging the strengths of two complementary models. Klein and Manning (2004) combine two complementary models for grammar induction, one that models constituency and one that models dependency, in a manner broadly similar to the current work. Aside from investigating a different domain, one novel aspect of this paper is that we present a formal objective and a training algorithm for combining two generic models.

6 Conclusion

We have described an efficient and fully unsupervised method of producing state-of-the-art word alignments. By training two simple sequence-based models to agree, we achieve substantial error reductions over standard models. Our jointly trained HMM models reduce AER by 29% over test-time intersected GIZA++ model 4 alignments and also increase our robustness to varying initialization regimens. While AER is only a weak indicator of final translation quality in many current translation systems, we hope that more accurate alignments can eventually lead to improvements in the end-to-end translation process.

Acknowledgments We thank the anonymous reviewers for their comments.

References

Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the COLT 1998*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19:263–311.

Michael Collins and Yoram Singer. 1999. Unsupervised Models for Named Entity Classification. In *Proceedings of EMNLP 1999*.

Dan Klein and Christopher D. Manning. 2004. Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. In *Proceedings of ACL 2004*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL 2003*.

E. Matusov, Zens. R., and H. Ney. 2004. Symmetric word alignments for statistical machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, August.

Robert C. Moore. 2004. Improving IBM Word Alignment Model 1. In *Proceedings of ACL 2004*.

Hermann Ney and Stephan Vogel. 1996. HMM-Based Word Alignment in Statistical Translation. In *COLING*.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29:19–51.

L. G. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201.

Appendix: Derivation of agreement EM

To simplify notation, we drop the explicit reference to the parameters θ. Lower bound the objective in Equation 3 by introducing a distribution $q(\mathbf{z};\mathbf{x})$ and using the concavity of $\log$:

$$\sum_{\mathbf{x}} \log p_1(\mathbf{x})p_2(\mathbf{x}) \sum_{\mathbf{z}} p_1(\mathbf{z} \mid \mathbf{x})p_2(\mathbf{z} \mid \mathbf{x}) \tag{4}$$

$$\geq \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{z};\mathbf{x}) \log \frac{p_1(\mathbf{x})p_2(\mathbf{x})p_1(\mathbf{z} \mid \mathbf{x})p_2(\mathbf{z} \mid \mathbf{x})}{q(\mathbf{z};\mathbf{x})} \tag{5}$$

$$= \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{z};\mathbf{x}) \log \frac{p_1(\mathbf{z} \mid \mathbf{x})p_2(\mathbf{z} \mid \mathbf{x})}{q(\mathbf{z};\mathbf{x})} + C \tag{6}$$

$$= \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{z};\mathbf{x}) \log p_1(\mathbf{x},\mathbf{z})p_2(\mathbf{x},\mathbf{z}) + D, \tag{7}$$

where C depends only on θ but not q and D depends only q but not θ. The E-step chooses q given a fixed θ to maximize the lower bound. Equation 6 is exactly $\sum_{\mathbf{x}} -\mathrm{KL}(q||p_1p_2) + C$, which is maximized by setting q proportional to p_1p_2. The M-step chooses θ given a fixed q. Equation 7 decomposes into two separate optimization problems.