A Discriminative Matching Approach to Word Alignment

Ben Taskar **Simon Lacoste-Julien** **Dan Klein**
Computer Science Division, EECS Department
University of California, Berkeley
Berkeley, CA 94720

Abstract

We present a discriminative, large-margin approach to feature-based matching for word alignment. In this framework, pairs of word tokens receive a matching score, which is based on features of that pair, including measures of association between the words, distortion between their positions, similarity of the orthographic form, and so on. Even with only 100 labeled training examples and simple features which incorporate counts from a large unlabeled corpus, we achieve AER performance close to IBM Model 4, in much less time. Including Model 4 predictions as features, we achieve a relative AER reduction of 22% in over intersected Model 4 alignments.

1 Introduction

The standard approach to word alignment from sentence-aligned bitexts has been to construct models which generate sentences of one language from the other, then fitting those generative models with EM (Brown et al., 1990; Och and Ney, 2003). This approach has two primary advantages and two primary drawbacks. In its favor, generative models of alignment are well-suited for use in a noisy-channel translation system. In addition, they can be trained in an unsupervised fashion, though in practice they do require labeled validation alignments for tuning model hyper-parameters, such as null counts or smoothing amounts, which are crucial to producing alignments of good quality. A primary drawback of the generative approach to alignment is that, as in all generative models, explicitly incorporating arbitrary features of the input is difficult. For example, when considering whether to align two words in the IBM models (Brown et al., 1990), one cannot easily include information about such features as orthographic similarity (for detecting cognates), presence of the pair in various dictionaries, similarity of the frequency of the two words, choices made by other alignment systems on this sentence pair, and so on. While clever models can implicitly capture some of these information sources, it takes considerable work, and can make the resulting models quite complex. A second drawback of generative translation models is that, since they are learned with EM, they require extensive processing of large amounts of data to achieve good performance. While tools like GIZA++ (Och and Ney, 2003) do make it easier to build on the long history of the generative IBM approach, they also underscore how complex high-performance generative models can, and have, become.

In this paper, we present a discriminative approach to word alignment. Word alignment is cast as a maximum weighted matching problem (Cormen et al., 1990) in which each pair of words (e_j, f_k) in a sentence pair (e, f) is associated with a score $s_{jk}(e, f)$ reflecting the desirability of the alignment of that pair. The alignment

for the sentence pair is then the highest scoring matching under some constraints, for example the requirement that matchings be one-to-one.

This view of alignment as graph matching is not, in itself, new: Melamed (2000) uses competitive linking to greedily construct matchings where the pair score is a measure of word-to-word association, and Matusov et al. (2004) find exact maximum matchings where the pair scores come from the alignment posteriors of generative models. Tiedemann (2003) proposes incorporating a variety of word association "clues" into a greedy linking algorithm.

What we contribute here is a principled approach for tractable and efficient learning of the alignment score $s_{jk}(e, f)$ as a function of arbitrary features of that token pair. This contribution opens up the possibility of doing the kind of feature engineering for alignment that has been so successful for other NLP tasks. We first present the algorithm for large margin estimation of the scoring function. We then show that our method can achieve AER rates comparable to unsymmetrized IBM Model 4, using extremely little labeled data (as few as 100 sentences) and a simple feature set. Remarkably, by including bi-directional IBM Model 4 predictions as features, we achieve an absolute AER of 5.4 on the English-French Hansards alignment task, a relative reduction of 22% in AER over intersected Model 4 alignments and, to our knowledge, the best AER result published on this task.

2 Algorithm

We model the alignment prediction task as a maximum weight bipartite matching problem, where nodes correspond to the words in the two sentences. For simplicity, we assume here that each word aligns to one or zero words in the other sentence. The edge weight s_{jk} represents the degree to which word j in one sentence can translate into the word k in the other sentence. Our goal is to find an alignment that maximizes the sum of edge scores. We represent a matching using a set of binary variables y_{jk} that are set to 1 if word j is assigned to word k in the other sentence, and 0 otherwise. The score of an assignment is the sum of edge scores: $s(\mathbf{y}) = \sum_{jk} s_{jk} y_{jk}$. The maximum weight bipartite matching problem, $\arg\max_{\mathbf{y} \in \mathcal{Y}} s(\mathbf{y})$, can be solved using well known combinatorial algorithms or the following linear program:

$$\max_{\mathbf{z}} \quad \sum_{jk} s_{jk} z_{jk} \tag{1}$$

$$\text{s.t.} \quad \sum_{j} z_{jk} \leq 1, \quad \sum_{k} z_{jk} \leq 1, \quad 0 \leq z_{jk} \leq 1,$$

where the continuous variables z_{jk} correspond to the binary variables y_{jk}. This LP is guaranteed to have integral (and hence optimal) solutions for any scoring function $s(\mathbf{y})$ (Schrijver, 2003). Note that although the above LP can be used to compute alignments, combinatorial algorithms are generally more efficient. However, we use the LP to develop the learning algorithm below.

For a sentence pair $\mathbf{x}$, we denote position pairs by $\mathbf{x}_{jk}$ and their scores as s_{jk}. We let $s_{jk} = \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$ for some user provided feature mapping $\mathbf{f}$ and abbreviate $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{jk} y_{jk} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$. We can include in the feature vector the identity of the two words, their relative positions in their respective sentences, their part-of-speech tags, their string similarity (for detecting cognates), and so on.

At this point, one can imagine estimating a linear matching model in multiple ways, including using conditional likelihood estimation, an averaged perceptron update (see which matchings are proposed and adjust the weights according to the difference between the guessed and target structures (Collins, 2002)), or in large-margin fashion. Conditional likelihood estimation using a log-linear model $P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}$ requires summing over all matchings to compute the normalization $Z_{\mathbf{w}}(\mathbf{x})$, which is #P-complete (Valiant, 1979). In our experiments, we therefore investigated the averaged perceptron in addition to the large-margin method outlined below.

2.1 Large-margin estimation

We follow the large-margin formulation of Taskar et al. (2005a). Our input is a set of training instances $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$, where each instance consists of a sentence pair $\mathbf{x}_i$ and a target

alignment $\mathbf{y}_i$. We would like to find parameters $\mathbf{w}$ that predict correct alignments on the training data:

$$\mathbf{y}_i \;=\; \arg\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \bar{\mathbf{y}}_i), \quad \forall i,$$

where $\mathcal{Y}_i$ is the space of matchings appropriate for the sentence pair i.

In standard classification problems, we typically measure the error of prediction, $\ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$, using the simple 0-1 loss. In structured problems, where we are jointly predicting multiple variables, the loss is often more complex. While the F-measure is a natural loss function for this task, we instead chose a sensible surrogate that fits better in our framework: Hamming distance between $\mathbf{y}_i$ and $\bar{\mathbf{y}}_i$, which simply counts the number of edges predicted incorrectly.

We use an SVM-like hinge upper bound on the loss $\ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$, given by $\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i}[\mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i)]$, where $\ell_i(\bar{\mathbf{y}}_i) = \ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$, and $\mathbf{f}_i(\bar{\mathbf{y}}_i) = \mathbf{f}(\mathbf{x}_i, \bar{\mathbf{y}}_i)$. Minimizing this upper bound encourages the true alignment $\mathbf{y}_i$ to be optimal with respect to $\mathbf{w}$ for each instance i:

$$\min_{||\mathbf{w}|| \leq \gamma} \sum_i \max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i),$$

where γ is a regularization parameter.

In this form, the estimation problem is a mixture of continuous optimization over $\mathbf{w}$ and combinatorial optimization over $\mathbf{y}_i$. In order to transform it into a more standard optimization problem, we need a way to efficiently handle the *loss-augmented inference,* $\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i}[\mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i)]$. This optimization problem has precisely the same form as the prediction problem whose parameters we are trying to learn — $\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i)$ — but with an additional term corresponding to the loss function. Our assumption that the loss function decomposes over the edges is crucial to solving this problem. In particular, we use weighted Hamming distance, which counts the number of variables in which a candidate solution $\bar{\mathbf{y}}_i$ differs from the target output $\mathbf{y}_i$, with different cost for false positives (c^+) and false negatives (c^-):

$$\begin{aligned}\ell_i(\bar{\mathbf{y}}_i) &= \sum_{jk} \left[c^- y_{i,jk}(1 - \bar{y}_{i,jk}) + c^+ \bar{y}_{i,jk}(1 - y_{i,jk})\right] \\ &= \sum_{jk} c^- y_{i,jk} + \sum_{jk} [c^+ - (c^- + c^+) y_{i,jk}] \bar{y}_{i,jk}.\end{aligned}$$

The loss-augmented matching problem can then be written as an LP similar to Equation 1 (without the constant term $\sum_{jk} c^- y_{i,jk}$):

$$\begin{aligned}\max_{\mathbf{z}} \quad & \sum_{jk} z_{i,jk} [\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{i,jk}) + c^+ - (c^- + c^+) y_{i,jk}] \\ \text{s.t.} \quad & \sum_j z_{i,jk} \leq 1, \quad \sum_k z_{i,jk} \leq 1, \quad 0 \leq z_{i,jk} \leq 1.\end{aligned}$$

Hence, without any approximations, we have a continuous optimization problem instead of a combinatorial one:

$$\max_{\bar{\mathbf{y}}_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\bar{\mathbf{y}}_i) + \ell_i(\bar{\mathbf{y}}_i) = d_i + \max_{\mathbf{z}_i \in \mathcal{Z}_i} (\mathbf{w}^\top \mathbf{F}_i + \mathbf{c}_i)^\top \mathbf{z}_i,$$

where $d_i = \sum_{jk} c^- y_{i,jk}$ is the constant term, $\mathbf{F}_i$ is the appropriate matrix that has a column of features $\mathbf{f}(\mathbf{x}_{i,jk})$ for each edge jk, $\mathbf{c}_i$ is the vector of the loss terms $c^+ - (c^- + c^+) y_{i,jk}$ and finally $\mathcal{Z}_i = \{\mathbf{z}_i : \sum_j z_{i,jk} \leq 1, \quad \sum_k z_{i,jk} \leq 1, \quad 0 \leq z_{i,jk} \leq 1\}$.

Plugging this LP back into our estimation problem, we have

$$\min_{||\mathbf{w}|| \leq \gamma} \max_{\mathbf{z} \in \mathcal{Z}} \quad \sum_i \mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i + \mathbf{c}_i^\top \mathbf{z}_i - \mathbf{w}^\top \mathbf{F}_i \mathbf{y}_i, \quad (2)$$

where $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$, $\mathcal{Z} = \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_m$. Instead of the derivation in Taskar et al. (2005a), which produces a joint convex optimization problem using Lagrangian duality, here we tackle the problem in its natural saddle-point form.

2.2 The extragradient method

For saddle-point problems, a well-known solution strategy is the extragradient method (Korpelevich, 1976), which is closely related to projected-gradient methods.

The gradient of the objective in Equation 2 is given by: $\sum_i \mathbf{F}_i(\mathbf{z}_i - \mathbf{y}_i)$ (with respect to $\mathbf{w}$) and $\mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i$ (with respect to each $\mathbf{z}_i$). We denote the Euclidean projection of a vector onto $\mathcal{Z}_i$ as $P_{\mathcal{Z}_i}(\mathbf{v}) = \arg\min_{\mathbf{u} \in \mathcal{Z}_i} ||\mathbf{v} - \mathbf{u}||$ and projection onto the ball $||\mathbf{w}|| \leq \gamma$ as $P_\gamma(\mathbf{w}) = \gamma \mathbf{w} / \max(\gamma, ||\mathbf{w}||)$.

An iteration of the extragradient method consists of two very simple steps, prediction:

$$\bar{\mathbf{w}}^{t+1} = P_\gamma(\mathbf{w}^t + \beta_k \sum_i \mathbf{F}_i(\mathbf{y}_i - \mathbf{z}_i^t));$$
$$\bar{\mathbf{z}}_i^{t+1} = P_{\mathcal{Z}_i}(\mathbf{z}_i^t + \beta_k(\mathbf{F}_i^\top \mathbf{w}^t + \mathbf{c}_i));$$

and correction:

$$\mathbf{w}^{t+1} = P_\gamma(\mathbf{w}^t + \beta_k \sum_i \mathbf{F}_i(\mathbf{y}_i - \bar{\mathbf{z}}_i^{t+1}));$$
$$\mathbf{z}_i^{t+1} = P_{\mathcal{Z}_i}(\mathbf{z}_i^t + \beta_k(\mathbf{F}_i^\top \bar{\mathbf{w}}^{t+1} + \mathbf{c}_i)),$$

where β_k are appropriately chosen step sizes. The method is guaranteed to converge linearly to a solution $\mathbf{w}^*, \mathbf{z}^*$ (Korpelevich, 1976; He and Liao, 2002; Taskar et al., 2005b). Please see `www.cs.berkeley.edu/~taskar/extragradient.pdf` for more details.

The key subroutine of the algorithm is Euclidean projection onto the feasible sets $\mathcal{Z}_i$. In case of word alignment, $\mathcal{Z}_i$ is the convex hull of bipartite matchings and the problem reduces to the much-studied minimum cost quadratic flow problem (Bertsekas et al., 1997). The projection problem $P_{\mathcal{Z}_i}(\mathbf{z}'_i)$ is given by

$$\min_{\mathbf{z}} \quad \sum_{jk} \frac{1}{2}(z'_{i,jk} - z_{i,jk})^2$$
$$\text{s.t.} \quad \sum_j z_{i,jk} \leq 1, \quad \sum_k z_{i,jk} \leq 1, \quad 0 \leq z_{i,jk} \leq 1.$$

We can now use a standard reduction of bipartite matching to min cost flow by introducing a source node connected to all the words in one sentence and a sink node connected to all the words in the other sentence, using edges of capacity 1 and cost 0. The original edges jk have a quadratic cost $\frac{1}{2}(z'_{i,jk} - z_{i,jk})^2$ and capacity 1. Now the minimum cost flow from the source to the sink computes projection of $\mathbf{z}'_i$ onto $\mathcal{Z}_i$ We use standard, publicly-available code for solving this problem (Guerriero and Tseng, 2002).

3 Experiments

We applied this matching algorithm to word-level alignment using the English-French Hansards data from the 2003 NAACL shared task (Mihalcea and Pedersen, 2003). This corpus consists of 1.1M automatically aligned sentences, and comes with a validation set of 39 sentence pairs 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:

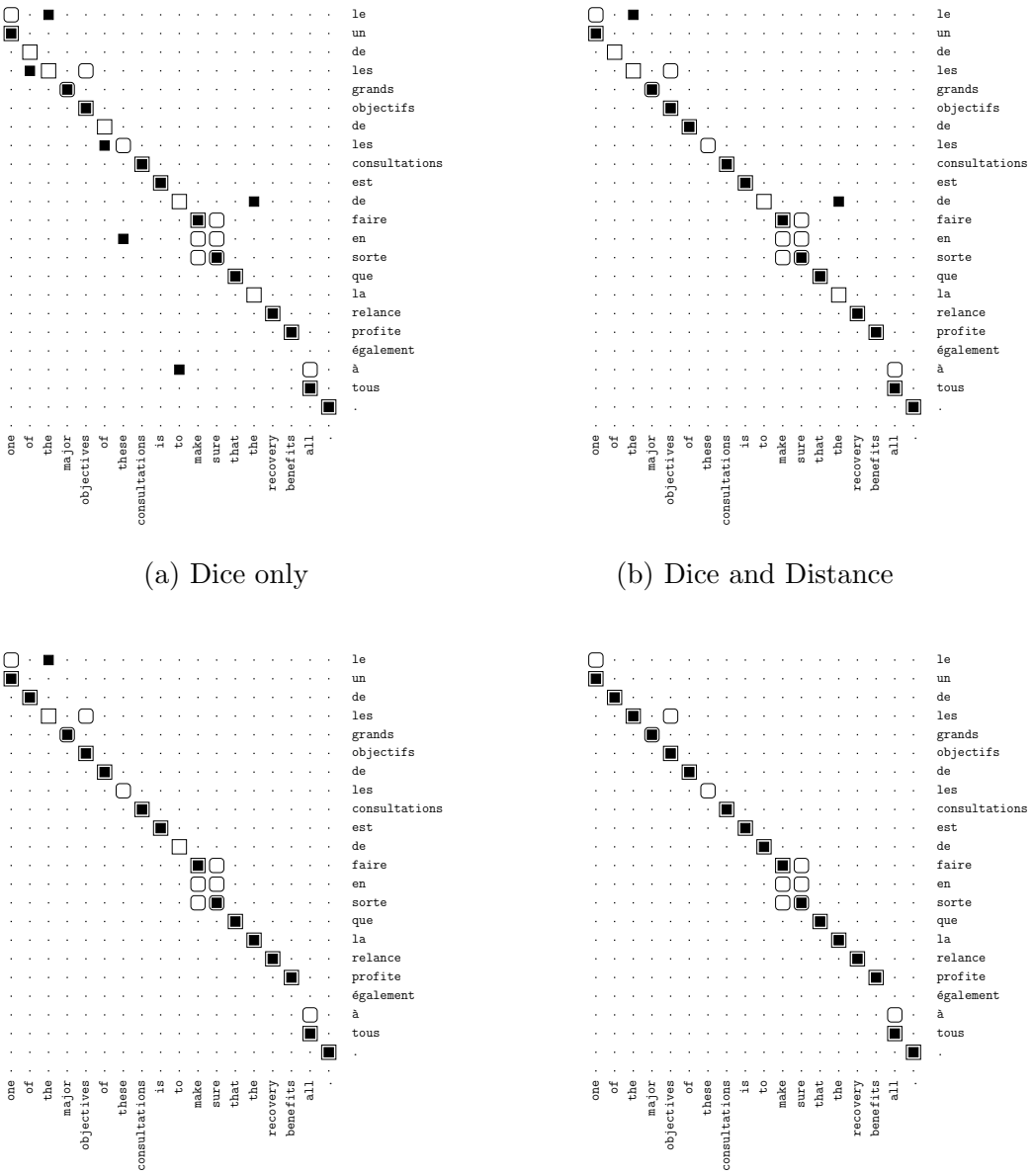
$$AER(A, S, P) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

Here, A is a set of proposed index pairs, S is the sure gold pairs, and P is the possible gold pairs. For example, in Figure 1, proposed alignments are shown against gold alignments, with open squares for sure alignments, rounded open squares for possible alignments, and filled black squares for proposed alignments.

Since our method is a supervised algorithm, we need labeled examples. For the training data, we split the original test set into 100 training examples and 347 test examples. In all our experiments, we used a structured loss function $\ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$ that penalized false negatives 3 times more than false positives, where 3 was picked by testing several values on the validation set. Instead of selecting a regularization parameter γ and running to convergence, we used early stopping as a cheap regularization method, by setting γ to a very large value (10000) and running the algorithm for 500 iterations. We selected a stopping point using the validation set by simply picking the best iteration on the validation set in terms of AER (ignoring the initial ten iterations, which were very noisy in our experiments). All selected iterations turned out to be in the first 50 iterations, as the algorithm converged fairly rapidly.

3.1 Features and Results

Very broadly speaking, the classic IBM models of word-level translation exploit four primary sources of knowledge and constraint: association of words (all IBM models), competition between alignments (all models), zero- or first-order preferences of alignment positions (2,4+), and fertility (3+). We model all of these in some way,



(a) Dice only

(b) Dice and Distance

(c) Dice, Distance, Orthographic, and BothShort

(d) All features

Figure 1: Example alignments for each successive feature set.

except fertility.[1]

First, and, most importantly, we want to include information about word association; translation pairs are likely to co-occur together in a bitext. This information can be captured, among many other ways, using a feature whose value is the Dice coefficient (Dice, 1945):

$$Dice(e, f) = \frac{2C_{EF}(e, f)}{C_E(e) + C_F(f)}$$

Here, C_E and C_F are counts of word occurrences in each language, while C_{EF} is the number of co-occurrences of the two words. With just this feature on a pair of word tokens (which depends only on their types), we can already make a stab

[1] In principle, we can model also model fertility, by allowing 0-k matches for each word rather than 0-1, and having bias features on each word. However, we did not explore this possibility.

at word alignment, aligning, say, each English word with the French word (or null) with the highest Dice value (see (Melamed, 2000)), simply as a matching-free heuristic model. With Dice counts taken from the 1.1M sentences, this gives and AER of 38.7 with English as the target, and 36.0 with French as the target (in line with the numbers from Och and Ney (2003)).

As observed in Melamed (2000), this use of Dice misses the crucial constraint of competition: a candidate source word with high association to a target word may be unavailable for alignment because some other target has an even better affinity for that source word. Melamed uses competitive linking to incorporate this constraint explicitly, while the IBM-style models get this effect via explaining-away effects in EM training. We can get something much like the combination of Dice and competitive linking by running with just one feature on each pair: the Dice value of that pair's words.[2] With just a Dice feature – meaning no learning is needed yet – we achieve an AER of 29.8, between the Dice with competitive linking result of 34.0 and Model 1 of 25.9 given in Och and Ney (2003). An example of the alignment at this stage is shown in Figure 1(a). Note that most errors lie off the diagonal, for example the often-correct *to-à* match.

IBM Model 2, as usually implemented, adds the preference of alignments to lie near the diagonal. Model 2 is driven by the product of a word-to-word measure and a (usually) Gaussian distribution which penalizes distortion from the diagonal. We can capture the same effect using features which reference the relative positions j and k of a pair (e_j, f_k). In addition to a Model 2-style quadratic feature referencing relative position, we threw in the following proximity features: absolute difference in relative position $abs(j/|e| - k/|f|)$, and the square and square root of this value. In addition, we used a conjunction feature of the dice coefficient times the proximity. Finally, we added a bias feature on each edge, which acts as a threshold that allows

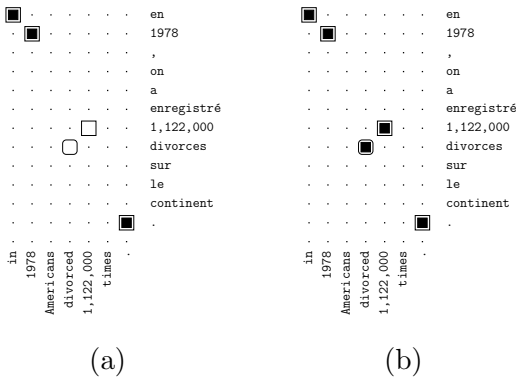


Figure 2: Example alignments showing the effects of orthographic cognate features. (a) Dice and Distance, (b) With Orthographic Features.

sparser, higher precision alignments. With these features, we got an AER of 15.5 (compare to 19.5 for Model 2 in (Och and Ney, 2003)). Note that we already have a capacity that Model 2 does not: we can learn a non-quadratic penalty with linear mixtures of our various components – this gives a similar effect to learning the variance of the Gaussian for Model 2, but is, at least in principle, more flexible.[3] These features fix the *to-à* error in Figure 1(a), giving the alignment in Figure 1(b).

On top of these features, we included other kinds of information, such as word-similarity features designed to capture cognate (and exact match) information. We added a feature for exact match of words, exact match ignoring accents, exact matching ignoring vowels, and fraction overlap of the longest common subsequence. Since these measures were only useful for long words, we also added a feature which indicates that both words in a pair are short. These orthographic and other features improved AER to 14.4. The running example now has the alignment in Figure 1(c), where one improvement may be attributable to the short pair feature – it has stopped proposing *the-de*, partially because the short pair feature downweights the score of that pair. A clearer example of these features making a difference is shown in Figure 2, where both the exact-match and character overlap fea-

[2]This isn't quite competitive linking, because we use a non-greedy matching.

[3]The learned response was in fact close to a Gaussian, but harsher near zero displacement.

tures are used.

One source of constraint which our model still does not explicitly capture is the first-order dependency between alignment positions, as in the HMM model (Vogel et al., 1996) and IBM models 4+. The *the-le* error in Figure 1(c) is symptomatic of this lack. In particular, it is a slightly better pair according to the Dice value than the correct *the-les*. However, the latter alignment has the advantage that *major-grands* follows it. To use this information source, we included a feature which gives the Dice value of the words following the pair.[4] We also added a word-frequency feature whose value is the absolute difference in log rank of the words, discouraging very common words from translating to very rare ones. Finally, we threw in bilexical features of the pairs of top 5 non-punctuation words in each language.[5] This helped by removing specific common errors like the residual tendency for French *de* to mistakenly align to English *the* (the two most common words). The resulting model produces the alignment in Figure 1(d). It has sorted out the *the-le / the-les* confusion, and is also able to guess *to-de*, which is not the most common translation for either word, but which is supported by the good Dice value on the following pair (*make-faire*).

With all these features, we got a final AER of 10.7, broadly similar to the 8.9 or 9.7 AERs of unsymmetrized IBM Model 4 trained on the same data that the Dice counts were taken from.[6] Of course, symmetrizing Model 4 by intersecting alignments from both directions does yield an improved AER of 6.9, so, while our model does do surprisingly well with cheaply obtained count-based features, Model 4 does still outperform it so far. However, our model can also easily incorporate the predictions of Model 4 as additional features. We therefore added three new features for each edge: the prediction of Model 4 in the English-French direction, the prediction in the French-English direction, and the intersection of the two predictions. With these powerful new features, our AER dropped dramatically to 5.4, a 22% improvement over the intersected Model 4 performance.

Model	AER
Dice (without matching)	38.7 / 36.0
Model 4 (E-F, F-E, intersected)	8.9 / 9.7/ 6.9
Discriminative Matching	
Dice Feature Only	29.8
+ Distance Features	15.5
+ Word Shape and Frequency	14.4
+ Common Words and Next-Dice	10.7
+ Model 4 Predictions	5.4

Figure 3: AER on the Hansards task.

Another way of doing the parameter estimation for this matching task would have been to use an averaged perceptron method, as in Collins (2002). In this method, we merely run our matching algorithm and update weights based on the difference between the predicted and target matchings. However, the performance of the average perceptron learner on the same feature set is much lower, only 8.1, not even breaking the AER of its best single feature (the intersected Model 4 predictions).

3.2 Scaling Experiments

We explored the scaling of our method by learning on a larger training set, which we created by using GIZA++ intersected bi-directional Model 4 alignments for the unlabeled sentence pairs. We then took the first 5K sentence pairs from these 1.1M Model 4 alignments. This gave us more training data, albeit with noisier labels. On a 3.4GHz Intel Xeon CPU, GIZA++ took 18 hours to align the 1.1M words, while our method learned its weights in between 6 minutes (100 training sentences) and three hours (5K sentences).

[4] It is important to note that while our matching algorithm has no first-order effects, the features can encode such effects in this way, or in better ways – e.g. using as features posteriors from the HMM model in the style of Matusov et al. (2004).

[5] The number of such features which can be learned depends on the number of training examples, and since some of our experiments used only a few dozen training examples we did not make heavy use of this feature.

[6] Note that the common word pair features affected common errors and therefore had a particularly large impact on AER.

4 Conclusions

We have presented a novel discriminative, large-margin method for learning word-alignment models on the basis of arbitrary features of word pairs. We have shown that our method is suitable for the common situation where a moderate number of good, fairly general features must be balanced on the basis of a small amount of labeled data. It is also likely that the method will be useful in conjunction with a large labeled alignment corpus (should such a set be created). We presented features capturing a few separate sources of information, producing alignments on the order of those given by unsymmetrized IBM Model 4 (using labeled training data of about the size others have used to tune generative models). In addition, when given bi-directional Model 4 predictions as features, our method provides a 22% AER reduction over intersected Model 4 predictions alone. The resulting 5.4 AER on the English-French Hansarks task is, to our knowledge, the best published AER figure for this training scenario (though since we use a subset of the test set, evaluations are not problem-free). Finally, our method scales to large numbers of training sentences and trains in minutes rather than hours or days for the higher-numbered IBM models, a particular advantage when not using features derived from those slower models.

References

D. P. Bertsekas, L. C. Polymenakos, and P. Tseng. 1997. An e-relaxation method for separable convex cost network flow problems. *SIAM J. Optim.*, 7(3):853–870.

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. 1990. *Introduction to Algorithms*. MIT Press, Cambridge, MA.

L. R. Dice. 1945. Measures of the amount of ecologic association between species. *Journal of Ecology*, 26:297–302.

F. Guerriero and P. Tseng. 2002. Implementation and test of auction methods for solving generalized network flow problems with separable convex cost. *Journal of Optimization Theory and Applications*, 115(1):113–144, October.

B.S. He and L. Z. Liao. 2002. Improvements of some projection methods for monotone nonlinear variational inequalities. *JOTA*, 112:111:128.

G. M. Korpelevich. 1976. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody*, 12:747:756.

E. Matusov, R. Zens, and H. Ney. 2004. Symmetric word alignments for statistical machine translation. In *Proc. of COLING 2004*.

I. D. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

R. Mihalcea and T. Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop, Building and Using parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–6, Edmonton, Alberta, Canada.

F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.

A. Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. 2005a. Learning structured prediction models: a large margin approach. In *Proceedings of the International Conference on Machine Learning*.

B. Taskar, S. Lacoste-Julien, and M. Jordan. 2005b. Structured prediction via the extragradient method. In *Proceedings of Neural Information Processing Systems*.

J. Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of EACL*.

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

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *COLING 16*, pages 836–841.