Hierarchical Search for Word Alignment

Jason Riesa and **Daniel Marcu**
Information Sciences Institute
Viterbi School of Engineering
University of Southern California
{riesa, marcu}@isi.edu

Abstract

We present a simple yet powerful hierarchical search algorithm for automatic word alignment. Our algorithm induces a forest of alignments from which we can efficiently extract a ranked *k*-best list. We score a given alignment within the forest with a flexible, linear discriminative model incorporating hundreds of features, and trained on a relatively small amount of annotated data. We report results on Arabic-English word alignment and translation tasks. Our model outperforms a GIZA++ Model-4 baseline by 6.3 points in F-measure, yielding a 1.1 BLEU score increase over a state-of-the-art syntax-based machine translation system.

1 Introduction

Automatic word alignment is generally accepted as a first step in training any statistical machine translation system. It is a vital prerequisite for generating translation tables, phrase tables, or syntactic transformation rules. Generative alignment models like IBM Model-4 (Brown et al., 1993) have been in wide use for over 15 years, and while not perfect (see Figure 1), they are completely unsupervised, requiring no annotated training data to learn alignments that have powered many current state-of-the-art translation system.

Today, there exist human-annotated alignments and an abundance of other information for many language pairs potentially useful for inducing accurate alignments. How can we take advantage of all of this data at our fingertips? Using feature functions that encode extra information is one good way. Unfortunately, as Moore (2005) points out, it is usually difficult to extend a given generative model with feature functions without changing the entire generative story. This difficulty has motivated much recent work in discriminative modeling for word alignment (Moore, 2005; Ittycheriah and Roukos, 2005; Liu et al., 2005; Taskar et al., 2005; Blunsom and Cohn, 2006; Lacoste-Julien et al., 2006; Moore et al., 2006).

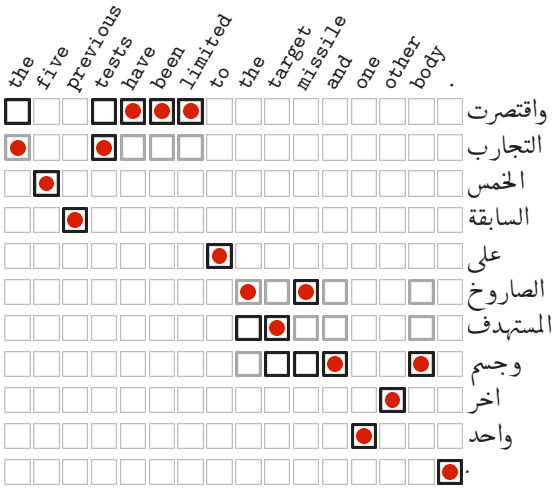


Figure 1: Model-4 alignment vs. a gold standard. Circles represent links in a human-annotated alignment, and black boxes represent links in the Model-4 alignment. Bold gray boxes show links gained after fully connecting the alignment.

We present in this paper a discriminative alignment model trained on relatively little data, with a simple, yet powerful hierarchical search procedure. We borrow ideas from both *k*-best parsing (Klein and Manning, 2001; Huang and Chiang, 2005; Huang, 2008) and forest-based, and hierarchical phrase-based translation (Huang and Chiang, 2007; Chiang, 2007), and apply them to word alignment.

Using a foreign string and an English parse tree as input, we formulate a bottom-up search on the parse tree, with the structure of the tree as a backbone for building a hypergraph of possible alignments. Our algorithm yields a forest of

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 157–166,
Uppsala, Sweden, 11-16 July 2010.

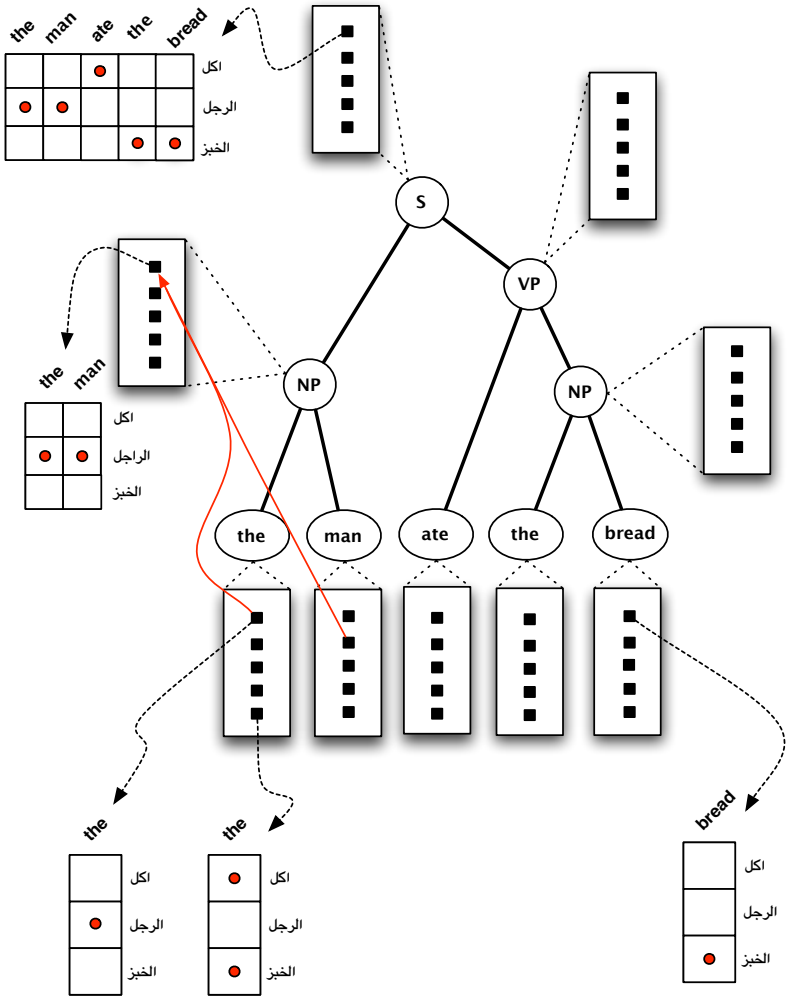


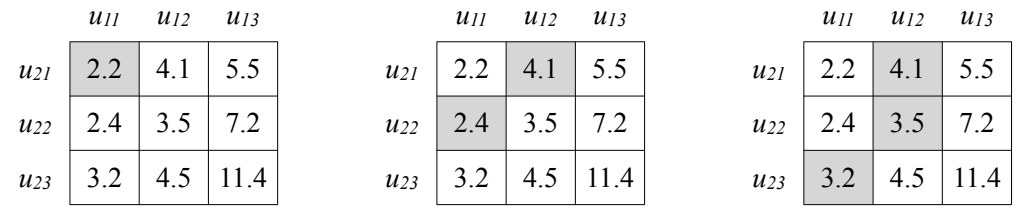
Figure 2: Example of approximate search through a hypergraph with beam size = 5. Each black square implies a partial alignment. Each partial alignment at each node is ranked according to its model score. In this figure, we see that the partial alignment implied by the 1-best hypothesis at the leftmost NP node is constructed by composing the best hypothesis at the terminal node labeled "the" and the 2nd-best hypothesis at the terminal node labeled "man". (We ignore terminal nodes in this toy example.) Hypotheses at the root node imply full alignment structures.

word alignments, from which we can efficiently extract the k-best. We handle an arbitrary number of features, compute them efficiently, and score alignments using a linear model. We train the parameters of the model using averaged perceptron (Collins, 2002) modified for structured outputs, but can easily fit into a max-margin or related framework. Finally, we use relatively little training data to achieve accurate word alignments. Our model can generate arbitrary alignments and learn from arbitrary gold alignments.

2 Word Alignment as a Hypergraph

Algorithm input The input to our alignment algorithm is a sentence-pair (e_1^n, f_1^m) and a parse tree over one of the input sentences. In this work, we parse our English data, and for each sentence $E = e_1^n$, let T be its syntactic parse. To generate parse trees, we use the Berkeley parser (Petrov et al., 2006), and use Collins head rules (Collins, 2003) to head-out binarize each tree.

Overview We present a brief overview here and delve deeper in Section 2.1. Word alignments are built bottom-up on the parse tree. Each node v in the tree holds *partial alignments* sorted by score.



(a) Score the left corner alignment first. Assume it is the 1-best. Numbers in the rest of the boxes are hidden at this point.

(b) Expand the frontier of alignments. We are now looking for the 2nd best.

(c) Expand the frontier further. After this step we have our top k alignments.

Figure 3: Cube pruning with alignment hypotheses to select the top-k alignments at node v with children $\langle u_1, u_2 \rangle$. In this example, $k = 3$. Each box represents the combination of two partial alignments to create a larger one. The score in each box is the sum of the scores of the child alignments plus a combination cost.

Each partial alignment comprises the columns of the alignment matrix for the e-words spanned by v, and each is scored by a linear combination of feature functions. See Figure 2 for a small example.

Initial partial alignments are enumerated and scored at preterminal nodes, each spanning a single column of the word alignment matrix. To speed up search, we can prune at each node, keeping a beam of size k. In the diagram depicted in Figure 2, the beam is size $k = 5$.

From here, we traverse the tree nodes bottom-up, combining partial alignments from child nodes until we have constructed a single full alignment at the root node of the tree. If we are interested in the k-best, we continue to populate the root node until we have k alignments.[1]

We use one set of feature functions for preterminal nodes, and another set for nonterminal nodes. This is analogous to local and nonlocal feature functions for parse-reranking used by Huang (2008). Using nonlocal features at a nonterminal node emits a combination cost for composing a set of child partial alignments.

Because combination costs come into play, we use cube pruning (Chiang, 2007) to approximate the k-best combinations at some nonterminal node v. Inference is exact when only local features are used.

Assumptions There are certain assumptions related to our search algorithm that we must make: (1) that using the structure of 1-best English syntactic parse trees is a reasonable way to frame and drive our search, and (2) that F-measure approximately decomposes over hyperedges.

We perform an oracle experiment to validate these assumptions. We find the oracle for a given (T,e,f) triple by proceeding through our search algorithm, forcing ourselves to always select correct links with respect to the gold alignment when possible, breaking ties arbitrarily. The the F_1 score of our oracle alignment is 98.8%, given this "perfect" model.

2.1 Hierarchical search

Initial alignments We can construct a word alignment hierarchically, bottom-up, by making use of the structure inherent in syntactic parse trees. We can think of building a word alignment as filling in an $M \times N$ matrix (Figure 1), and we begin by visiting each preterminal node in the tree. Each of these nodes spans a single e word. (Line 2 in Algorithm 1).

From here we can assign links from each e word to zero or more f words (Lines 6–14). At this level of the tree the span size is 1, and the partial alignment we have made spans a single column of the matrix. We can make many such partial alignments depending on the links selected. Lines 5 through 9 of Algorithm 1 enumerate either the null alignment, single-link alignments, or two-link alignments. Each partial alignment is scored and stored in a sorted heap (Lines 9 and 13).

In practice enumerating all two-link alignments can be prohibitive for long sentence pairs; we set a practical limit and score only pairwise combina-

[1] We use approximate dynamic programming to store alignments, keeping only scored lists of pointers to initial single-column spans. Each item in the list is a derivation that implies a partial alignment.

Algorithm 1: Hypergraph Alignment

Input:
Source sentence e_1^n
Target sentence f_1^m
Parse tree T over e_1^n
Set of feature functions $\mathbf{h}$
Weight vector $\mathbf{w}$
Beam size k

Output:
A k-best list of alignments over e_1^n and f_1^m

1 **function** ALIGN(e_1^n, f_1^m, T)
2 **for** $v \in T$ *in bottom-up order* **do**
3 $\alpha_v \leftarrow \emptyset$
4 **if** IS-PRETERMINALNODE(v) **then**
5 $i \leftarrow$ index-of(v)
6 **for** $j = 0$ **to** m **do**
7 $links \leftarrow (i, j)$
8 $score \leftarrow \mathbf{w} \cdot \mathbf{h}(links, v, e_1^n, f_1^m)$
9 PUSH(α_v, $\langle score, links \rangle$, k)
10 **for** $k = j + 1$ **to** m **do**
11 $links \leftarrow (i, j), (i, k)$
12 $score \leftarrow \mathbf{w} \cdot \mathbf{h}(links, v, e_1^n, f_1^m)$
13 PUSH(α_v, $\langle score, links \rangle$, k)
14 **end**
15 **end**
16 **else**
17 $\alpha_v \leftarrow$ GROWSPAN(children(v), k)
18 **end**
19 **end**
20 **end**
21 **function** GROWSPAN($\langle u_1, u_2 \rangle, k$)
22 **return** CUBEPRUNING($\langle \alpha_{u_1}, \alpha_{u_2} \rangle, k$,**w**,**h**)
23 **end**

tions of the top $n = \max\left\{\frac{|f|}{2}, 10\right\}$ scoring single-link alignments.

We limit the number of total partial alignments α_v kept at each node to k. If at any time we wish to push onto the heap a new partial alignment when the heap is full, we pop the current worst off the heap and replace it with our new partial alignment if its score is better than the current worst.

Building the hypergraph We now visit internal nodes (Line 16) in the tree in bottom-up order. At each nonterminal node v we wish to combine the partial alignments of its children $u_1, \ldots, u_c$. We use cube pruning (Chiang, 2007; Huang and Chiang, 2007) to select the k-best combinations of the partial alignments of $u_1, \ldots, u_c$ (Line 19). Note

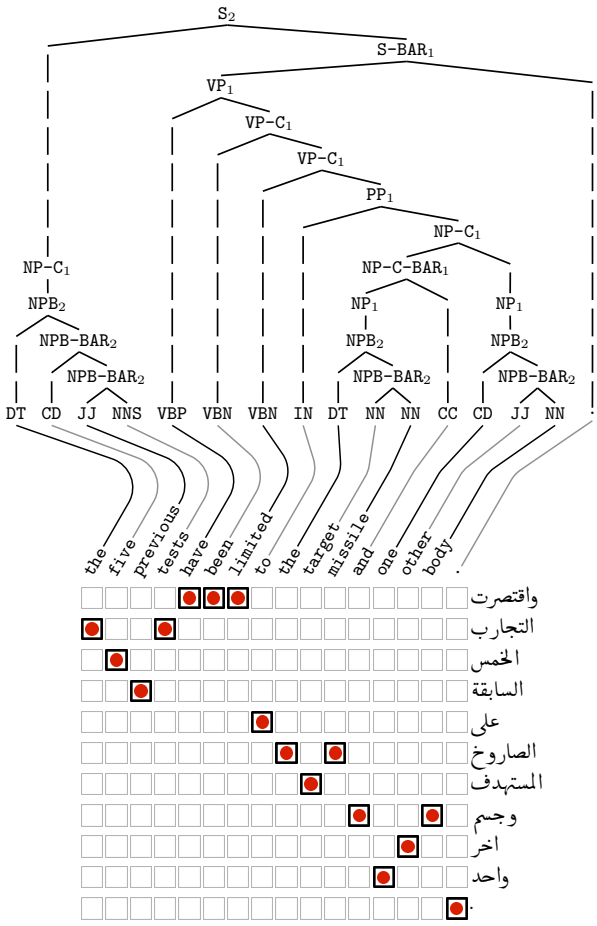


Figure 4: Correct version of Figure 1 after hypergraph alignment. Subscripts on the nonterminal labels denote the branch containing the head word for that span.

that Algorithm 1 assumes a binary tree[2], but is not necessary. In the general case, cube pruning will operate on a d-dimensional hypercube, where d is the branching factor of node v.

We cannot enumerate and score every possibility; without the cube pruning approximation, we will have k^c possible combinations at each node, exploding the search space exponentially. Figure 3 depicts how we select the top-k alignments at a node v from its children $\langle u_1, u_2 \rangle$.

3 Discriminative training

We incorporate all our new features into a linear model and learn weights for each using the online averaged perceptron algorithm (Collins, 2002) with a few modifications for structured outputs inspired by Chiang et al. (2008). We define:

[2]We find empirically that using binarized trees reduces search errors in cube pruning.

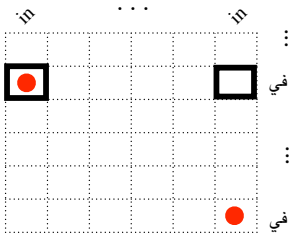


Figure 5: A common problem with GIZA++ Model 4 alignments is a weak distortion model. The second English "in" is aligned to the wrong Arabic token. Circles show the gold alignment.

$$\gamma(y) = \ell(y_i, y) + \mathbf{w} \cdot (\mathbf{h}(y_i) - \mathbf{h}(y)) \quad (1)$$

where $\ell(y_i,y)$ is a loss function describing how bad it is to guess y when the correct answer is y_i. In our case, we define $\ell(y_i,y)$ as $1-F_1(y_i,y)$. We select the oracle alignment according to:

$$y^+ = \arg\min_{y \in \text{CAND}(x)} \gamma(y) \quad (2)$$

where $\text{CAND}(x)$ is a set of hypothesis alignments generated from input x. Instead of the traditional oracle, which is calculated solely with respect to the loss $\ell(y_i,y)$, we choose the oracle that jointly minimizes the loss and the difference in model score to the true alignment. Note that Equation 2 is equivalent to maximizing the sum of the F-measure and model score of y:

$$y^+ = \arg\max_{y \in \text{CAND}(x)} (F_1(y_i, y) + \mathbf{w} \cdot \mathbf{h}(y)) \quad (3)$$

Let $\hat{y}$ be the 1-best alignment according to our model:

$$\hat{y} = \arg\max_{y \in \text{CAND}(x)} \mathbf{w} \cdot \mathbf{h}(y) \quad (4)$$

Then, at each iteration our weight update is:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(\mathbf{h}(y^+) - \mathbf{h}(\hat{y})) \quad (5)$$

where η is a learning rate parameter.[3] We find that this more conservative update gives rise to a much more stable search. After each iteration, we expect y^+ to get closer and closer to the true y_i.

4 Features

Our simple, flexible linear model makes it easy to throw in many features, mapping a given complex alignment structure into a single high-dimensional feature vector. Our hierarchical search framework allows us to compute these features when needed, and affords us extra useful syntactic information.

We use two classes of features: *local* and *non-local*. Huang (2008) defines a feature h to be local if and only if it can be factored among the local productions in a tree, and non-local otherwise. Analogously for alignments, our class of local features are those that can be factored among the local partial alignments competing to comprise a larger span of the matrix, and non-local otherwise. These features score a set of links and the words connected by them.

Feature development Our features are inspired by analysis of patterns contained among our gold alignment data and automatically generated parse trees. We use both local lexical and nonlocal structural features as described below.

4.1 Local features

These features fire on single-column spans.

- From the output of GIZA++ Model 4, we compute **lexical probabilities** $p(e \mid f)$ and $p(f \mid e)$, as well as a **fertility table** $\phi(e)$. From the fertility table, we fire features $\phi_0(e)$, $\phi_1(e)$, and $\phi_{2+}(e)$ when a word e is aligned to zero, one, or two or more words, respectively. Lexical probability features $p(e \mid f)$ and $p(f \mid e)$ fire when a word e is aligned to a word f.

- Based on these features, we include a binary **lexical-zero** feature that fires if both $p(e \mid f)$ and $p(f \mid e)$ are equal to zero for a given word pair (e, f). Negative weights essentially penalize alignments with links never seen before in the Model 4 alignment, and positive weights encourage such links. We employ a separate instance of this feature for each English part-of-speech tag: $p(f \mid e, t)$.

  We learn a different feature weight for each. Critically, this feature tells us how much to trust alignments involving nouns, verbs, adjectives, function words, punctuation, etc. from the Model 4 alignments from which our $p(e \mid f)$ and $p(f \mid e)$ tables are built. Table 1 shows a sample of learned weights. Intuitively, alignments involving English parts-of-speech more likely to be content words (e.g. NNPS, NNS, NN) are more trustworthy

[3] We set η to 0.05 in our experiments.

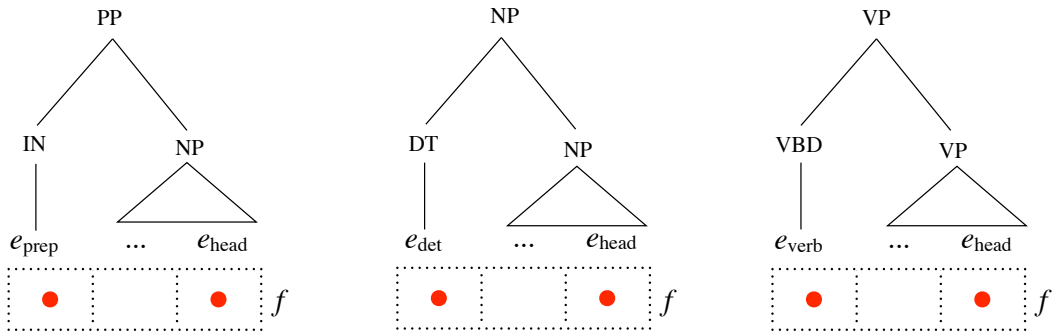


Figure 6: Features PP-NP-head, NP-DT-head, and VP-VP-head fire on these tree-alignment patterns. For example, PP-NP-head fires exactly when the head of the PP is aligned to exactly the same f words as the head of it's sister NP.

	Penalty
NNPS	−1.11
NNS	−1.03
NN	−0.80
NNP	−0.62
VB	−0.54
VBG	−0.52
JJ	−0.50
JJS	−0.46
VBN	−0.45
⋮	⋮
POS	−0.0093
EX	−0.0056
RP	−0.0037
WP$	−0.0011
TO	0.037
	Reward

Table 1: A sampling of learned weights for the **lexical zero** feature. Negative weights penalize links never seen before in a baseline alignment used to initialize lexical $p(e \mid f)$ and $p(f \mid e)$ tables. Positive weights outright reward such links.

than those likely to be function words (e.g. TO, RP, EX), where the use of such words is often radically different across languages.

- We also include a measure of **distortion**. This feature returns the distance to the diagonal of the matrix for any link in a partial alignment. If there is more than one link, we return the distance of the link farthest from the diagonal.

- As a lexical backoff, we include a **tag probability** feature, $p(t \mid f)$ that fires for some link (e, f) if the part-of-speech tag of e is t. The conditional probabilities in this table are computed from our parse trees and the baseline Model 4 alignments.

- In cases where the lexical probabilities are too strong for the distortion feature to overcome (see Figure 5), we develop the **multiple-distortion** feature. Although local features do not know the partial alignments at other spans, they do have access to the entire English sentence at every step because our input is constant. If some e exists more than once in e_1^n we fire this feature on all links containing word e, returning again the distance to the diagonal for that link. We learn a strong negative weight for this feature.

- We find that binary **identity** and **punctuation-mismatch** features are important. The binary identity feature fires if $e = f$, and proves useful for untranslated numbers, symbols, names, and punctuation in the data. Punctuation-mismatch fires on any link that causes nonpunctuation to be aligned to punctuation.

Additionally, we include fine-grained versions of the lexical probability, fertility, and distortion features. These fire for for each link (e, f) and part-of-speech tag. That is, we learn a separate weight for each feature for each part-of-speech tag in our data. Given the tag of e, this affords the model the ability to pay more or less attention to the features described above depending on the tag given to e.

Arabic-English specific features We describe here language specific features we implement to exploit shallow Arabic morphology.

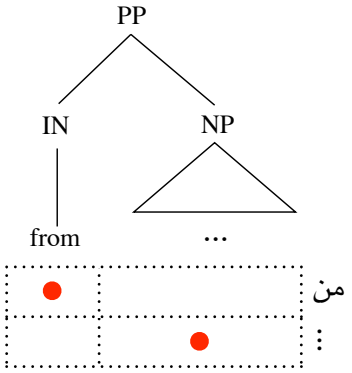


Figure 7: This figure depicts the tree/alignment structure for which the feature **PP-from-prep** fires. The English preposition "from" is aligned to Arabic word من. Any aligned words in the span of the sister NP are aligned to words following من. English preposition structure commonly matches that of Arabic in our gold data. This family of features captures these observations.

- We observe the Arabic prefix و, transliterated `w-` and generally meaning *and*, to prepend to most any word in the lexicon, so we define features $p_{\neg w}(e \mid f)$ and $p_{\neg w}(f \mid e)$. If f begins with `w-`, we strip off the prefix and return the values of $p(e \mid f)$ and $p(f \mid e)$. Otherwise, these features return 0.
- We also include analogous feature functions for several functional and pronominal prefixes and suffixes.[4]

4.2 Nonlocal features

These features comprise the combination cost component of a partial alignment score and may fire when concatenating two partial alignments to create a larger span. Because these features can look into any two arbitrary subtrees, they are considered nonlocal features as defined by Huang (2008).

- Features **PP-NP-head**, **NP-DT-head**, and **VP-VP-head** (Figure 6) all exploit headwords on the parse tree. We observe English prepositions and determiners to often align to the headword of their sister. Likewise, we observe the head of a VP to align to the head of an immediate sister VP.

In Figure 4, when the search arrives at the left-most NPB node, the NP-DT-head feature will fire given this structure and links over the span `[the ... tests]`. When search arrives at the second NPB node, it will fire given the structure and links over the span `[the ... missle]`, but will not fire at the right-most NPB node.

- Local lexical preference features compete with the headword features described above. However, we also introduce nonlocal lexicalized features for the most common types of English and foreign prepositions to also compete with these general headword features.

  PP features **PP-of-prep**, **PP-from-prep**, **PP-to-prep**, **PP-on-prep**, and **PP-in-prep** fire at any PP whose left child is a preposition and right child is an NP. The head of the PP is one of the enumerated English prepositions and is aligned to any of the three most common foreign words to which it has also been observed aligned in the gold alignments. The last constraint on this pattern is that all words under the span of the sister NP, if aligned, must align to words following the foreign preposition. Figure 7 illustrates this pattern.

- Finally, we have a **tree-distance** feature to avoid making too many many-to-one (from many English words to a single foreign word) links. This is a simplified version of and similar in spirit to the tree distance metric used in (DeNero and Klein, 2007). For any pair of links (e_i, f) and (e_j, f) in which the e words differ but the f word is the same token in each, return the tree height of first common ancestor of e_i and e_j.

  This feature captures the intuition that it is much worse to align two English words at different ends of the tree to the same foreign word, than it is to align two English words under the same NP to the same foreign word.

  To see why a string distance feature that counts only the flat horizontal distance from e_i to e_j is not the best strategy, consider the following. We wish to align a determiner to the same f word as its sister head noun under the same NP. Now suppose there are several intermediate adjectives separating the determiner and noun. A string distance met-

[4] Affixes used by our model are currently: بـ, لـ, الـ, بالـ, ـي, ـكم, ـكما, ـهم, ـهما. Others either we did not experiment with, or seemed to provide no significant benefit, and are not included.

ric, with no knowledge of the relationship between determiner and noun will levy a much heavier penalty than its tree distance analog.

5 Related Work

Recent work has shown the potential for syntactic information encoded in various ways to support inference of superior word alignments. Very recent work in word alignment has also started to report downstream effects on BLEU score.

Cherry and Lin (2006) introduce soft syntactic ITG (Wu, 1997) constraints into a discriminative model, and use an ITG parser to constrain the search for a Viterbi alignment. Haghighi et al. (2009) confirm and extend these results, showing BLEU improvement for a hierarchical phrase-based MT system on a small Chinese corpus. As opposed to ITG, we use a linguistically motivated phrase-structure tree to drive our search and inform our model. And, unlike ITG-style approaches, our model can generate arbitrary alignments and learn from arbitrary gold alignments.

DeNero and Klein (2007) refine the distortion model of an HMM aligner to reflect tree distance instead of string distance. Fossum et al. (2008) start with the output from GIZA++ Model-4 union, and focus on increasing precision by deleting links based on a linear discriminative model exposed to syntactic and lexical information.

Fraser and Marcu (2007) take a semi-supervised approach to word alignment, using a small amount of gold data to further tune parameters of a headword-aware generative model. They show a significant improvement over a Model-4 union baseline on a very large corpus.

6 Experiments

We evaluate our model and and resulting alignments on Arabic-English data against those induced by IBM Model-4 using GIZA++ (Och and Ney, 2003) with both the union and grow-diag-final heuristics. We use 1,000 sentence pairs and gold alignments from LDC2006E86 to train model parameters: 800 sentences for training, 100 for testing, and 100 as a second held-out development set to decide when to stop perceptron training. We also align the test data using GIZA++[5] along with 50 million words of English.

[5]We use a standard training procedure: 5 iterations of Model-1, 5 iterations of HMM, 3 iterations of Model-3, and 3 iterations of Model-4.

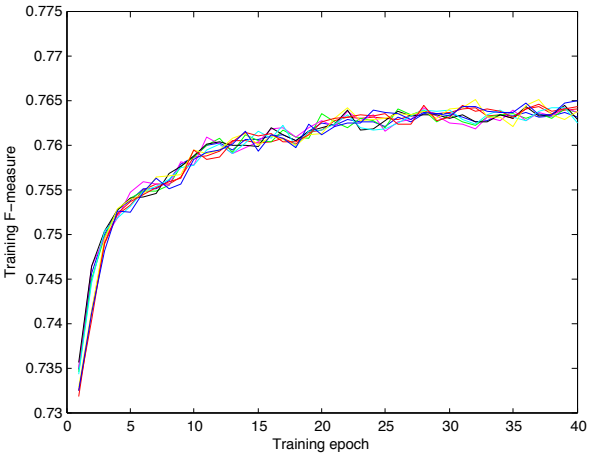


Figure 8: Learning curves for 10 random restarts over time for parallel averaged perceptron training. These plots show the current F-measure on the training set as time passes. Perceptron training here is quite stable, converging to the same general neighborhood each time.

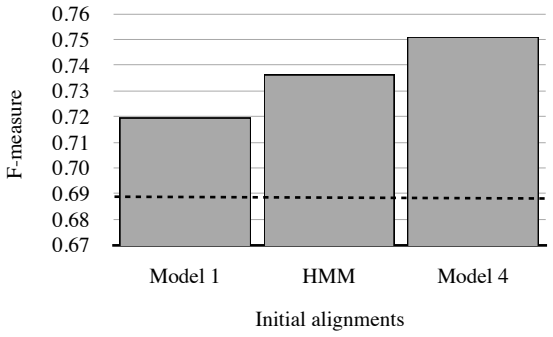


Figure 9: Model robustness to the initial alignments from which the $p(e \mid f)$ and $p(f \mid e)$ features are derived. The dotted line indicates the baseline accuracy of GIZA++ Model 4 alone.

6.1 Alignment Quality

We empirically choose our beam size k from the results of a series of experiments, setting k=1, 2, 4, 8, 16, 32, and 64. We find setting $k = 16$ to yield the highest accuracy on our held-out test data. Using wider beams results in higher F-measure on training data, but those gains do not translate into higher accuracy on held-out data.

The first three columns of Table 2 show the balanced F-measure, Precision, and Recall of our alignments versus the two GIZA++ Model-4 baselines. We report an F-measure 8.6 points over Model-4 union, and 6.3 points over Model-4 grow-diag-final.

	F	P	R	Arabic/English BLEU	# Unknown Words
M4 (union)	.665	.636	.696	45.1	2,538
M4 (grow-diag-final)	.688	.702	.674	46.4	2,262
Hypergraph alignment	**.751**	**.780**	**.724**	**47.5**	**1,610**

Table 2: F-measure, Precision, Recall, the resulting BLEU score, and number of unknown words on a held-out test corpus for three types of alignments. BLEU scores are case-insensitive IBM BLEU. We show a 1.1 BLEU increase over the strongest baseline, Model-4 grow-diag-final. This is statistically significant at the $p < 0.01$ level.

Figure 8 shows the stability of the search procedure over ten random restarts of parallel averaged perceptron training with 40 CPUs. Training examples are randomized at each epoch, leading to slight variations in learning curves over time but all converge into the same general neighborhood.

Figure 9 shows the robustness of the model to initial alignments used to derive lexical features $p(e \mid f)$ and $p(f \mid e)$. In addition to IBM Model 4, we experiment with alignments from Model 1 and the HMM model. In each case, we significantly outperform the baseline GIZA++ Model 4 alignments on a heldout test set.

6.2 MT Experiments

We align a corpus of 50 million words with GIZA++ Model-4, and extract translation rules from a 5.4 million word core subset. We align the same core subset with our trained hypergraph alignment model, and extract a second set of translation rules. For each set of translation rules, we train a machine translation system and decode a held-out test corpus for which we report results below.

We use a syntax-based translation system for these experiments. This system transforms Arabic strings into target English syntax trees Translation rules are extracted from (e-tree, f-string, alignment) triples as in (Galley et al., 2004; Galley et al., 2006).

We use a randomized language model (similar to that of Talbot and Brants (2008)) of 472 million English words. We tune the the parameters of the MT system on a held-out development corpus of 1,172 parallel sentences, and test on a held-out parallel corpus of 746 parallel sentences. Both corpora are drawn from the NIST 2004 and 2006 evaluation data, with no overlap at the document or segment level with our training data.

Columns 4 and 5 in Table 2 show the results of our MT experiments. Our hypergraph alignment algorithm allows us a 1.1 BLEU increase over the best baseline system, Model-4 grow-diag-final. This is statistically significant at the $p < 0.01$ level. We also report a 2.4 BLEU increase over a system trained with alignments from Model-4 union.

7 Conclusion

We have opened up the word alignment task to advances in hypergraph algorithms currently used in parsing and machine translation decoding. We treat word alignment as a parsing problem, and by taking advantage of English syntax and the hypergraph structure of our search algorithm, we report significant increases in both F-measure and BLEU score over standard baselines in use by most state-of-the-art MT systems today.

Acknowledgements

We would like to thank our colleagues in the Natural Language Group at ISI for many meaningful discussions and the anonymous reviewers for their thoughtful suggestions. This research was supported by DARPA contract HR0011-06-C-0022 under subcontract to BBN Technologies, and a USC CREATE Fellowship to the first author.

References

Phil Blunsom and Trevor Cohn. 2006. Discriminative Word Alignment with Conditional Random Fields. In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

Peter F. Brown, Stephen A. Della Pietra, Vincent Della J. Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312. MIT Press. Camrbidge, MA. USA.

Colin Cherry and Dekang Lin. 2006. Soft Syntactic Constraints for Word Alignment through Discriminative Training. In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*. 33(2):201–228. MIT Press. Cambridge, MA. USA.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online Large-Margin Training of Syntactic and Structural Translation Features. In *Proceedings of EMNLP*. Honolulu, HI. USA.

Michael Collins. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*. 29(4):589–637. MIT Press. Cambridge, MA. USA.

Michael Collins 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

John DeNero and Dan Klein. 2007. Tailoring Word Alignments to Syntactic Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL*. Prague, Czech Republic.

Alexander Fraser and Daniel Marcu. 2007. Getting the Structure Right for Word Alignment: LEAF. In *Proceedings of EMNLP-CoNLL*. Prague, Czech Republic.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using Syntax to Improve Word Alignment Precision for Syntax-Based Machine Translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*. Columbus, Ohio.

Dan Klein and Christopher D. Manning. 2001. Parsing and Hypergraphs. In *Proceedings of the 7th International Workshop on Parsing Technologies*. Beijing, China.

Aria Haghighi, John Blitzer, and Dan Klein. 2009. Better Word Alignments with Supervised ITG Models. In *Proceedings of ACL-IJCNLP 2009*. Singapore.

Liang Huang and David Chiang. 2005. Better *k*-best Parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies*. Vancouver, BC. Canada.

Liang Huang and David Chiang. 2007. Forest Rescoring: Faster Decoding with Integrated Language Models. In *Proceedings of the 45th Annual Meeting of the ACL*. Prague, Czech Republic.

Liang Huang. 2008. Forest Reranking: Discriminative Parsing with Non-Local Features. In *Proceedings of the 46th Annual Meeting of the ACL*. Columbus, OH. USA.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a Translation Rule? In *Proceedings of NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable Inference and Training of Context-Rich Syntactic Models In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of HLT-EMNLP*. Vancouver, BC. Canada.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I. Jordan. 2006. Word alignment via Quadratic Assignment. In *Proceedings of HLT-EMNLP*. New York, NY. USA.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear Models for Word Alignment In *Proceedings of the 43rd Annual Meeting of the ACL*. Ann Arbor, Michigan. USA.

Robert C. Moore. 2005. A Discriminative Framework for Word Alignment. In *Proceedings of EMNLP*. Vancouver, BC. Canada.

Robert C. Moore, Wen-tau Yih, and Andreas Bode. 2006. Improved Discriminative Bilingual Word Alignment In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*. 29(1):19–52. MIT Press. Cambridge, MA. USA.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein 2006. Learning Accurate, Compact, and Interpretable Tree Annotation In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

Kishore Papineni, Salim Roukos, T. Ward, and W-J. Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation In *Proceedings of the 40th Annual Meeting of the ACL*. Philadelphia, PA. USA.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A Discriminative Matching Approach to Word Alignment. In *Proceedings of HLT-EMNLP*. Vancouver, BC. Canada.

David Talbot and Thorsten Brants. 2008. Randomized Language Models via Perfect Hash Functions. In *Proceedings of ACL-08: HLT*. Columbus, OH. USA.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*. 23(3):377–404. MIT Press. Cambridge, MA. USA.