A Constrained Viterbi Relaxation for Bidirectional Word Alignment

Yin-Wen Chang **Alexander M. Rush**
MIT CSAIL,
Cambridge, MA 02139
{yinwen,srush}@
csail.mit.edu

John DeNero
UC Berkeley,
Berkeley, CA 94720
denero@
cs.berkeley.edu

Michael Collins
Columbia University,
New York, NY 10027
mcollins@
cs.columbia.edu

Abstract

Bidirectional models of word alignment are an appealing alternative to post-hoc combinations of directional word aligners. Unfortunately, most bidirectional formulations are NP-Hard to solve, and a previous attempt to use a relaxation-based decoder yielded few exact solutions (6%). We present a novel relaxation for decoding the bidirectional model of DeNero and Macherey (2011). The relaxation can be solved with a modified version of the Viterbi algorithm. To find optimal solutions on difficult instances, we alternate between incrementally adding constraints and applying optimality-preserving coarse-to-fine pruning. The algorithm finds provably exact solutions on 86% of sentence pairs and shows improvements over directional models.

1 Introduction

Word alignment is a critical first step for building statistical machine translation systems. In order to ensure accurate word alignments, most systems employ a post-hoc symmetrization step to combine directional word aligners, such as IBM Model 4 (Brown et al., 1993) or hidden Markov model (HMM) based aligners (Vogel et al., 1996). Several authors have proposed bidirectional models that incorporate this step directly, but decoding under many bidirectional models is NP-Hard and finding exact solutions has proven difficult.

In this paper, we describe a novel Lagrangian-relaxation based decoder for the bidirectional model proposed by DeNero and Macherey (2011), with the goal of improving search accuracy. In that work, the authors implement a dual decomposition-based decoder for the problem, but are only able to find exact solutions for around 6% of instances.

Our decoder uses a simple variant of the Viterbi algorithm for solving a relaxed version of this model. The algorithm makes it easy to re-introduce constraints for difficult instances, at the cost of increasing run-time complexity. To offset this cost, we employ optimality-preserving coarse-to-fine pruning to reduce the search space. The pruning method utilizes lower bounds on the cost of valid bidirectional alignments, which we obtain from a fast, greedy decoder.

The method has the following properties:

- It is based on a novel relaxation for the model of DeNero and Macherey (2011), solvable with a variant of the Viterbi algorithm.
- To find optimal solutions, it employs an efficient strategy that alternates between adding constraints and applying pruning.
- Empirically, it is able to find exact solutions on 86% of sentence pairs and is significantly faster than general-purpose solvers.

We begin in Section 2 by formally describing the directional word alignment problem. Section 3 describes a preliminary bidirectional model using full agreement constraints and a Lagrangian relaxation-based solver. Section 4 modifies this model to include adjacency constraints. Section 5 describes an extension to the relaxed algorithm to explicitly enforce constraints, and Section 6 gives a pruning method for improving the efficiency of the algorithm.

Experiments compare the search error and accuracy of the new bidirectional algorithm to several directional combiners and other bidirectional algorithms. Results show that the new relaxation is much more effective at finding exact solutions and is able to produce comparable alignment accuracy.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1481–1490,
Baltimore, Maryland, USA, June 23-25 2014.

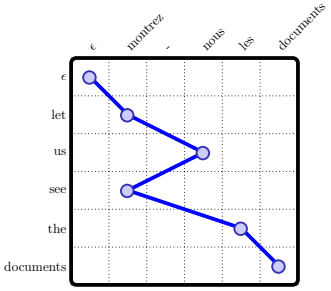


Figure 1: An example $\mathbf{e}\rightarrow\mathbf{f}$ directional alignment for the sentences `let us see the documents` and `montrez - nous les documents`, with $I = 5$ and $J = 5$. The indices $i \in [I]_0$ are rows, and the indices $j \in [J]_0$ are columns. The HMM alignment shown has transitions $x(0,1,1) = x(1,2,3) = x(3,3,1) = x(1,4,4) = x(4,5,5) = 1$.

Notation We use lower- and upper-case letters for scalars and vectors, and script-case for sets e.g. $\mathcal{X}$. For vectors, such as $v \in \{0,1\}^{(\mathcal{I}\times\mathcal{J})\cup\mathcal{J}}$, where $\mathcal{I}$ and $\mathcal{J}$ are finite sets, we use the notation $v(i,j)$ and $v(j)$ to represent elements of the vector. Define $d = \delta(i)$ to be the indicator vector with $d(i) = 1$ and $d(i') = 0$ for all $i' \neq i$. Finally define the notation $[J]$ to refer to $\{1 \dots J\}$ and $[J]_0$ to refer to $\{0 \dots J\}$.

2 Background

The focus of this work is on the word alignment decoding problem. Given a sentence $\mathbf{e}$ of length $|\mathbf{e}| = I$ and a sentence $\mathbf{f}$ of length $|\mathbf{f}| = J$, our goal is to find the best bidirectional alignment between the two sentences under a given objective function. Before turning to the model of interest, we first introduce directional word alignment.

2.1 Word Alignment

In the $\mathbf{e}\rightarrow\mathbf{f}$ word alignment problem, each word in $\mathbf{e}$ is aligned to a word in $\mathbf{f}$ or to the null word ϵ. This alignment is a mapping from each index $i \in [I]$ to an index $j \in [J]_0$ (where $j = 0$ represents alignment to ϵ). We refer to a single word alignment as a *link*.

A first-order HMM alignment model (Vogel et al., 1996) is an HMM of length $I + 1$ where the hidden state at position $i \in [I]_0$ is the aligned index $j \in [J]_0$, and the transition score takes into account the previously aligned index $j' \in [J]_0$.[1] Formally, define the set of possible HMM alignments as $\mathcal{X} \subset \{0,1\}^{([I]_0\times[J]_0)\cup([I]\times[J]_0\times[J]_0)}$ with

$$\mathcal{X} = \begin{cases} x : x(0,0) = 1, \\ x(i,j) = \sum_{j'=0}^{J} x(j',i,j) & \forall i \in [I], j \in [J]_0, \\ x(i,j) = \sum_{j'=0}^{J} x(j,i+1,j') & \forall i \in [I-1]_0, j \in [J]_0 \end{cases}$$

where $x(i,j) = 1$ indicates that there is a link between index i and index j, and $x(j',i,j) = 1$ indicates that index $i-1$ aligns to index j' and index i aligns to j. Figure 1 shows an example member of $\mathcal{X}$.

The constraints of $\mathcal{X}$ enforce *backward* and *forward* consistency respectively. If $x(i,j) = 1$, backward consistency enforces that there is a transition from $(i-1,j')$ to (i,j) for some $j' \in [J]_0$, whereas forward consistency enforces a transition from (i,j) to $(i+1,j')$ for some $j' \in [J]_0$. Informally the constraints "chain" together the links.

The HMM objective function $f : \mathcal{X} \rightarrow \mathbb{R}$ can be written as a linear function of x

$$f(x;\theta) = \sum_{i=1}^{I}\sum_{j=0}^{J}\sum_{j'=0}^{J} \theta(j',i,j)x(j',i,j)$$

where the vector $\theta \in \mathbb{R}^{[I]\times[J]_0\times[J]_0}$ includes the transition and alignment scores. For a generative model of alignment, we might define $\theta(j',i,j) = \log(p(\mathbf{e}_i|\mathbf{f}_j)p(j|j'))$. For a discriminative model of alignment, we might define $\theta(j',i,j) = w \cdot \phi(i,j',j,\mathbf{f},\mathbf{e})$ for a feature function ϕ and weights w (Moore, 2005; Lacoste-Julien et al., 2006).

Now reverse the direction of the model and consider the $\mathbf{f}\rightarrow\mathbf{e}$ alignment problem. An $\mathbf{f}\rightarrow\mathbf{e}$ alignment is a binary vector $y \in \mathcal{Y}$ where for each $j \in [J]$, $y(i,j) = 1$ for exactly one $i \in [I]_0$. Define the set of HMM alignments $\mathcal{Y} \subset \{0,1\}^{([I]_0\times[J]_0)\cup([I]_0\times[I]_0\times[J])}$ as

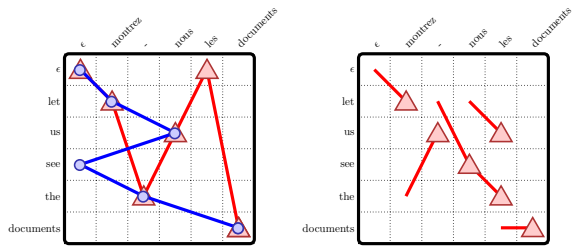
$$\mathcal{Y} = \begin{cases} y : y(0,0) = 1, \\ y(i,j) = \sum_{i'=0}^{I} y(i',i,j) & \forall i \in [I]_0, j \in [J], \\ y(i,j) = \sum_{i'=0}^{I} y(i,i',j+1) & \forall i \in [I]_0, j \in [J-1]_0 \end{cases}$$

Similarly define the objective function

$$g(y;\omega) = \sum_{j=1}^{J}\sum_{i=0}^{I}\sum_{i'=0}^{I} \omega(i',i,j)y(i',i,j)$$

with vector $\omega \in \mathbb{R}^{[I]_0\times[I]_0\times[J]}$.

[1] Our definition differs slightly from other HMM-based aligners in that it does not track the last ϵ alignment.



(a) (b)

Figure 2: (a) An example alignment pair (x, y) satisfying the full agreement conditions. The x alignment is represented with circles and the y alignment with triangles. (b) An example $\mathbf{f}\rightarrow\mathbf{e}$ alignment $y \in \mathcal{Y}'$ with relaxed forward constraints. Note that unlike an alignment from $\mathcal{Y}$ multiple words may be aligned in a column and words may transition from non-aligned positions.

Note that for both of these models we can solve the optimization problem exactly using the standard Viterbi algorithm for HMM decoding. The first can be solved in $O(IJ^2)$ time and the second in $O(I^2J)$ time.

3 Bidirectional Alignment

The directional bias of the $\mathbf{e}\rightarrow\mathbf{f}$ and $\mathbf{f}\rightarrow\mathbf{e}$ alignment models may cause them to produce differing alignments. To obtain the best single alignment, it is common practice to use a post-hoc algorithm to merge these directional alignments (Och et al., 1999). First, a directional alignment is found from each word in $\mathbf{e}$ to a word $\mathbf{f}$. Next an alignment is produced in the reverse direction from $\mathbf{f}$ to $\mathbf{e}$. Finally, these alignments are merged, either through intersection, union, or with an interpolation algorithm such as grow-diag-final (Koehn et al., 2003).

In this work, we instead consider a bidirectional alignment model that jointly considers both directional models. We begin in this section by introducing a simple bidirectional model that enforces *full* agreement between directional models and giving a relaxation for decoding. Section 4 loosens this model to *adjacent* agreement.

3.1 Enforcing Full Agreement

Perhaps the simplest post-hoc merging strategy is to retain the intersection of the two directional models. The analogous bidirectional model enforces *full* agreement to ensure the two alignments select the same non-null links i.e.

$$x^*, y^* = \arg\max_{x\in\mathcal{X}, y\in\mathcal{Y}} f(x) + g(y) \text{ s.t.}$$
$$x(i,j) = y(i,j) \quad \forall i \in [I], j \in [J]$$

We refer to the optimal alignments for this problem as x^* and y^*.

Unfortunately this bidirectional decoding model is NP-Hard (a proof is given in Appendix A). As it is common for alignment pairs to have $|\mathbf{f}|$ or $|\mathbf{e}|$ over 40, exact decoding algorithms are intractable in the worst-case.

Instead we will use Lagrangian relaxation for this model. At a high level, we will remove a subset of the constraints from the original problem and replace them with Lagrange multipliers. If we can solve this new problem efficiently, we may be able to get optimal solutions to the original problem. (See the tutorial by Rush and Collins (2012) describing the method.)

There are many possible subsets of constraints to consider relaxing. The relaxation we use preserves the agreement constraints while relaxing the Markov structure of the $\mathbf{f}\rightarrow\mathbf{e}$ alignment. This relaxation will make it simple to later re-introduce constraints in Section 5.

We relax the forward constraints of set $\mathcal{Y}$. Without these constraints the y links are no longer chained together. This has two consequences: (1) for index j there may be any number of indices i, such that $y(i,j) = 1$, (2) if $y(i',i,j) = 1$ it is no longer required that $y(i', j-1) = 1$. This gives a set $\mathcal{Y}'$ which is a superset of $\mathcal{Y}$

$$\mathcal{Y}' = \left\{ \begin{array}{l} y : y(0,0) = 1, \\ y(i,j) = \sum_{i'=0}^{I} y(i',i,j) \quad \forall i \in [I]_0, j \in [J] \end{array} \right.$$

Figure 2(b) shows a possible $y \in \mathcal{Y}'$ and a valid unchained structure.

To form the Lagrangian dual with relaxed forward constraints, we introduce a vector of Lagrange multipliers, $\lambda \in \mathbb{R}^{[I-1]_0 \times [J]_0}$, with one multiplier for each original constraint. The Lagrangian dual $L(\lambda)$ is defined as

$$\max_{\substack{x\in\mathcal{X}, y\in\mathcal{Y}', \\ x(i,j)=y(i,j)}} f(x) + \sum_{i=1}^{I}\sum_{j=0}^{J}\sum_{i'=0}^{I} y(i',i,j)\omega(i',i,j) \quad (1)$$
$$- \sum_{i=0}^{I}\sum_{j=0}^{J-1} \lambda(i,j)\left(y(i,j) - \sum_{i'=0}^{I} y(i,i',j+1)\right)$$
$$= \max_{\substack{x\in\mathcal{X}, y\in\mathcal{Y}', \\ x(i,j)=y(i,j)}} f(x) + \sum_{i=1}^{I}\sum_{j=0}^{J}\sum_{i'=0}^{I} y(i',i,j)\omega'(i',i,j) \quad (2)$$
$$= \max_{\substack{x\in\mathcal{X}, y\in\mathcal{Y}', \\ x(i,j)=y(i,j)}} f(x) + \sum_{i=1}^{I}\sum_{j=0}^{J} y(i,j) \max_{i'\in[I]_0} \omega'(i',i,j) \quad (3)$$
$$= \max_{\substack{x\in\mathcal{X}, y\in\mathcal{Y}', \\ x(i,j)=y(i,j)}} f(x) + g'(y;\omega,\lambda) \quad (4)$$

Line 2 distributes the λ's and introduces a modified potential vector ω' defined as

$$\omega'(i', i, j) = \omega(i', i, j) - \lambda(i, j) + \lambda(i', j-1)$$

for all $i' \in [I]_0, i \in [I]_0, j \in [J]$. Line 3 utilizes the relaxed set $\mathcal{Y}'$ which allows each $y(i,j)$ to select the best possible previous link $(i', j-1)$. Line 4 introduces the modified directional objective

$$g'(y; \omega, \lambda) = \sum_{i=1}^{I} \sum_{j=0}^{J} y(i,j) \max_{i' \in [I]_0} \omega'(i', i, j)$$

The Lagrangian dual is guaranteed to be an upper bound on the optimal solution, i.e. for all λ, $L(\lambda) \geq f(x^*) + g(y^*)$. Lagrangian relaxation attempts to find the tighest possible upper bound by minimizing the Lagrangian dual, $\min_\lambda L(\lambda)$, using subgradient descent. Briefly, subgradient descent is an iterative algorithm, with two steps. Starting with $\lambda = 0$, we iteratively

1. Set (x, y) to the $\arg\max$ of $L(\lambda)$.
2. Update $\lambda(i,j)$ for all $i \in [I-1]_0, j \in [J]_0$,

$$\lambda(i,j) \leftarrow \lambda(i,j) - \eta_t \big(y(i,j) - \sum_{i'=0}^{I} y(i, i', j+1)\big)$$

.

where $\eta_t > 0$ is a step size for the t'th update. If at any iteration of the algorithm the forward constraints are satisfied for (x, y), then $f(x) + g(y) = f(x^*) + g(x^*)$ and we say this gives a *certificate of optimality* for the underlying problem.

To run this algorithm, we need to be able to efficiently compute the (x, y) pair that is the $\arg\max$ of $L(\lambda)$ for any value of λ. Fortunately, since the y alignments are no longer constrained to valid transitions, we can compute these alignments by first picking the best $\mathbf{f} \rightarrow \mathbf{e}$ transitions for each possible link, and then running an $\mathbf{e} \rightarrow \mathbf{f}$ Viterbi-style algorithm to find the bidirectional alignment.

The $\max$ version of this algorithm is shown in Figure 3. It consists of two steps. We first compute the score for each $y(i,j)$ variable. We then use the standard Viterbi update for computing the x variables, adding in the score of the $y(i,j)$ necessary to satisfy the constraints.

procedure VITERBIFULL(θ, ω')
  Let π, ρ be dynamic programming charts.
  $\rho[i,j] \leftarrow \max_{i' \in [I]_0} \omega'(i', i, j) \; \forall \, i \in [I], j \in [J]_0$
  $\pi[0,0] \leftarrow \sum_{j=1}^{J} \max\{0, \rho[0,j]\}$
  for $i \in [I], j \in [J]_0$ in order **do**
    $\pi[i,j] \leftarrow \max_{j' \in [J]_0} \theta(j', i, j) + \pi[i-1, j']$
    if j $\neq 0$ **then** $\pi[i,j] \leftarrow \pi[i,j] + \rho[i,j]$
  return $\max_{j \in [J]_0} \pi[I,j]$

Figure 3: Viterbi-style algorithm for computing $L(\lambda)$. For simplicity the algorithm shows the $\max$ version of the algorithm, $\arg\max$ can be computed with back-pointers.

4 Adjacent Agreement

Enforcing full agreement can be too strict an alignment criteria. DeNero and Macherey (2011) instead propose a model that allows near matches, which we call *adjacent* agreement. Adjacent agreement allows links from one direction to agree with adjacent links from the reverse alignment for a small penalty. Figure 4(a) shows an example of a valid bidirectional alignment under adjacent agreement.

In this section we formally introduce adjacent agreement, and propose a relaxation algorithm for this model. The key algorithmic idea is to extend the Viterbi algorithm in order to consider possible adjacent links in the reverse direction.

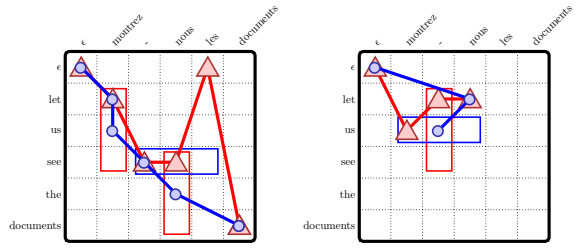
4.1 Enforcing Adjacency

Define the adjacency set $\mathcal{K} = \{-1, 0, 1\}$. A bidirectional alignment satisfies adjacency if for all $i \in [I], j \in [J]$,

- If $x(i,j) = 1$, it is required that $y(i+k, j) = 1$ for exactly one $k \in \mathcal{K}$ (i.e. either above, center, or below). We indicate which position with variables $z^{\updownarrow}_{i,j} \in \{0,1\}^{\mathcal{K}}$
- If $x(i,j) = 1$, it is allowed that $y(i, j+k) = 1$ for any $k \in \mathcal{K}$ (i.e. either left, center, or right) and all other $y(i, j') = 0$. We indicate which positions with variables $z^{\leftrightarrow}_{i,j} \in \{0,1\}^{\mathcal{K}}$

Formally for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the pair (x, y) is feasible if there exists a z from the set $\mathcal{Z}(x,y) \subset \{0,1\}^{\mathcal{K}^2 \times [I] \times [J]}$ defined as

$$\mathcal{Z}(x,y) = \left\{ \begin{array}{ll} z : \forall i \in [I], j \in [J] & \\ z^{\updownarrow}_{i,j} \in \{0,1\}^{\mathcal{K}}, & z^{\leftrightarrow}_{i,j} \in \{0,1\}^{\mathcal{K}} \\ x(i,j) = \sum_{k \in \mathcal{K}} z^{\updownarrow}_{i,j}(k), & \sum_{k \in \mathcal{K}} z^{\leftrightarrow}_{i,j}(k) = y(i,j), \\ z^{\updownarrow}_{i,j}(k) \leq y(i+k, j) & \forall k \in \mathcal{K} : i + k > 0, \\ x(i,j) \geq z^{\leftrightarrow}_{i,j-k}(k) & \forall k \in \mathcal{K} : j + k > 0 \end{array} \right.$$



(a) (b)

Figure 4: (a) An alignment satisfying the adjacency constraints. Note that $x(2,1) = 1$ is allowed because of $y(1,1) = 1$, $x(4,3) = 1$ because of $y(3,3)$, and $y(3,1)$ because of $x(3,2)$. (b) An adjacent bidirectional alignment in progress. Currently $x(2,2) = 1$ with $z^{\updownarrow}(-1) = 1$ and $z^{\leftrightarrow}(-1) = 1$. The last transition was from $x(1,3)$ with $z^{\leftrightarrow\prime}(-1) = 1, z^{\leftrightarrow\prime}(0) = 1, z^{\updownarrow\prime}(0) = 1$.

Additionally adjacent, non-overlapping matches are assessed a penalty α calculated as

$$h(z) = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k \in \mathcal{K}} \alpha |k| (z^{\updownarrow}_{i,j}(k) + z^{\leftrightarrow}_{i,j}(k))$$

where $\alpha \leq 0$ is a parameter of the model. The example in Figure 4(a) includes a 3α penalty.

Adding these penalties gives the complete adjacent agreement problem

$$\arg\max_{\substack{z \in \mathcal{Z}(x,y) \\ x \in \mathcal{X}, y \in \mathcal{Y}}} f(x) + g(y) + h(z)$$

Next, apply the same relaxation from Section 3.1, i.e. we relax the forward constraints of the $\mathbf{f} \rightarrow \mathbf{e}$ set. This yields the following Lagrangian dual

$$L(\lambda) = \max_{\substack{z \in \mathcal{Z}(x,y) \\ x \in \mathcal{X}, y \in \mathcal{Y}'}} f(x) + g'(y; \omega, \lambda) + h(z)$$

Despite the new constraints, we can still compute $L(\lambda)$ in $O(IJ(I+J))$ time using a variant of the Viterbi algorithm. The main idea will be to consider possible adjacent settings for each link. Since each $z^{\updownarrow}_{i,j}$ and $z^{\leftrightarrow}_{i,j}$ only have a constant number of settings, this does not increase the asymptotic complexity of the algorithm.

Figure 5 shows the algorithm for computing $L(\lambda)$. The main loop of the algorithm is similar to Figure 3. It proceeds row-by-row, picking the best alignment $x(i,j) = 1$. The major change is that the chart π also stores a value $z \in \{0,1\}^{\mathcal{K} \times \mathcal{K}}$ representing a possible $z^{\updownarrow}_{i,j}, z^{\leftrightarrow}_{i,j}$ pair. Since we have

procedure VITERBIADJ(θ, ω')
$\rho[i,j] \leftarrow \max_{i' \in [I]_0} \omega'(i', i, j) \; \forall \, i \in [I], j \in [J]_0$
$\pi[0,0] \leftarrow \sum_{j=1}^{J} \max\{0, \rho[0,j]\}$
for $i \in [I], j \in [J]_0, z^{\updownarrow}, z^{\leftrightarrow} \in \{0,1\}^{|\mathcal{K}|}$ **do**
$\pi[i,j,z] \leftarrow$
$\max_{\substack{j' \in [J]_0, \\ z' \in \mathcal{N}(z, j-j')}} \theta(j', i, j) + \pi[i-1, j', z']$
$+ \sum_{k \in \mathcal{K}} z^{\leftrightarrow}(k)(\rho[i, j+k] + \alpha|k|)$
$+ z^{\updownarrow}(k)\alpha|k|$
return $\max_{j \in [J]_0, z \in \{0,1\}^{|\mathcal{K}| \times |\mathcal{K}|}} \pi[I, j, z]$

Figure 5: Modified Viterbi algorithm for computing the adjacent agreement $L(\lambda)$.

the proposed $z_{i,j}$ in the inner loop, we can include the scores of the adjacent y alignments that are in neighboring columns, as well as the possible penalty for matching $x(i,j)$ to a $y(i+k,j)$ in a different row. Figure 4(b) gives an example setting of z.

In the dynamic program, we need to ensure that the transitions between the z's are consistent. The vector z' indicates the y links adjacent to $x(i-1, j')$. If j' is near to j, z' may overlap with z and vice-versa. The transition set $\mathcal{N}$ ensures these indicators match up

$$\mathcal{N}(z, k') = \left\{ z' : \begin{array}{l} (z^{\updownarrow}(-1) \wedge k' \in \mathcal{K}) \Rightarrow z^{\leftrightarrow\prime}(k'), \\ (z^{\updownarrow\prime}(1) \wedge k' \in \mathcal{K}) \Rightarrow z^{\leftrightarrow}(-k'), \\ \sum_{k \in \mathcal{K}} z^{\updownarrow}(k) = 1 \end{array} \right.$$

5 Adding Back Constraints

In general, it can be shown that Lagrangian relaxation is only guaranteed to solve a linear programming relaxation of the underlying combinatorial problem. For difficult instances, we will see that this relaxation often does not yield provably exact solutions. However, it is possible to "tighten" the relaxation by re-introducing constraints from the original problem.

In this section, we extend the algorithm to allow incrementally re-introducing constraints. In particular we track which constraints are most often violated in order to explicitly enforce them in the algorithm.

Define a binary vector $p \in \{0,1\}^{[I-1]_0 \times [J]_0}$ where $p(i,j) = 1$ indicates a previously relaxed constraint on link $y(i,j)$ that should be reintroduced into the problem. Let the new partially

constrained Lagrangian dual be defined as

$$L(\lambda; p) = \max_{\substack{z \in \mathcal{Z}(x,y) \\ x \in \mathcal{X}, y \in \mathcal{Y}'}} f(x) + g'(y; \omega, \lambda) + h(z)$$

$$y(i,j) = \sum_{i'} y(i, i', j+1) \quad \forall i, j : p(i,j) = 1$$

If $p = \vec{1}$, the problem includes all of the original constraints, whereas $p = \vec{0}$ gives our original Lagrangian dual. In between we have progressively more constrained variants.

In order to compute the $\arg\max$ of this optimization problem, we need to satisfy the constraints within the Viterbi algorithm. We augment the Viterbi chart with a count vector $d \in \mathcal{D}$ where $\mathcal{D} \subset \mathbb{Z}^{||p||_1}$ and $d(i,j)$ is a count for the (i,j)'th constraint, i.e. $d(i,j) = y(i,j) - \sum_{i'} y(i', i, j)$. Only solutions with count 0 at the final position satisfy the active constraints. Additionally define a helper function $[\cdot]_{\mathcal{D}}$ as the projection from $\mathbb{Z}^{[I-1]_0 \times [J]} \rightarrow \mathcal{D}$, which truncates dimensions without constraints.

Figure 6 shows this constrained Viterbi relaxation approach. It takes p as an argument and enforces the active constraints. For simplicity, we show the full agreement version, but the adjacent agreement version is similar. The main new addition is that the inner loop of the algorithm ensures that the count vector d is the sum of the counts of its children d' and $d - d'$.

Since each additional constraint adds a dimension to d, adding constraints has a multiplicative impact on running time. Asymptotically the new algorithm requires $O(2^{||p||_1} IJ(I+J))$ time. This is a problem in practice as even adding a few constraints can make the problem intractable. We address this issue in the next section.

procedure CONSVITERBIFULL(θ, ω', p)
 for $i \in [I], j \in [J]_0, i' \in [I]$ **do**
  $d \leftarrow |\delta(i,j) - \delta(i', j-1)|_{\mathcal{D}}$
  $\rho[i,j,d] \leftarrow \omega'(i', i, j)$
 for $j \in [J], d \in \mathcal{D}$ **do**
  $\pi[0,0,d] \leftarrow \max_{d' \in \mathcal{D}} \pi[0,0,d'] + \rho[0,j,d-d']$
 for $i \in [I], j \in [J]_0, d \in \mathcal{D}$ **do**
  if j = 0 **then**
   $\pi[i,j,d] \leftarrow \max_{j' \in [J]_0} \theta(j', i, j) + \pi[i-1, j', d]$
  else
   $\pi[i,j,d] \leftarrow \max_{j' \in [J]_0, d' \in \mathcal{D}} \theta(j', i, j) + \pi[i-1, j', d'] + \rho[i, j, d-d']$
 return $\max_{j \in [J]_0} \pi[I, j, \mathbf{0}]$

Figure 6: Constrained Viterbi algorithm for finding partially-constrained, full-agreement alignments. The argument p indicates which constraints to enforce.

6 Pruning

Re-introducing constraints can lead to an exponential blow-up in the search space of the Viterbi algorithm. In practice though, many alignments in this space are far from optimal, e.g. aligning a common word like `the` to `nous` instead of `les`. Since Lagrangian relaxation re-computes the alignment many times, it would be preferable to skip these links in later rounds, particularly after re-introducing constraints.

In this section we describe an optimality preserving coarse-to-fine algorithm for pruning. Approximate coarse-to-fine pruning algorithms are widely used within NLP, but exact pruning is less common. Our method differs in that it only eliminates non-optimal transitions based on a lower-bound score. After introducing the pruning method, we present an algorithm to make this method effective in practice by producing high-scoring lower bounds for adjacent agreement.

6.1 Thresholding Max-Marginals

Our pruning method is based on removing transitions with low max-marginal values. Define the max-marginal value of an $\mathbf{e} \rightarrow \mathbf{f}$ transition in our Lagrangian dual as

$$M(j', i, j; \lambda) = \max_{\substack{z \in \mathcal{Z}(x,y), \\ x \in \mathcal{X}, y \in \mathcal{Y}'}} f(x) + g'(y; \lambda) + h(z) \quad \text{s.t. } x(j', i, j) = 1$$

where M gives the value of the best dual alignment that transitions from $(i-1, j')$ to (i, j). These max-marginals can be computed by running a forward-backward variant of any of the algorithms described thus far.

We make the following claim about max-marginal values and any lower-bound score

Lemma 1 (Safe Pruning). *For any valid constrained alignment $x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}(x,y)$ and for any dual vector $\lambda \in \mathbb{R}^{[I-1]_0 \times [J]_0}$, if there exists a transition j', i, j with max-marginal value $M(j', i, j; \lambda) < f(x) + g(y) + h(z)$ then the transition will not be in the optimal alignment, i.e. $x^*(j', i, j) = 0$.*

This lemma tells us that we can prune transitions whose dual max-marginal value falls below

a threshold without pruning possibly optimal transitions. Pruning these transitions can speed up Lagrangian relaxation without altering its properties.

Furthermore, the threshold is determined by any feasible lower bound on the optimal score, which means that better bounds can lead to more pruning.

6.2 Finding Lower Bounds

Since the effectiveness of pruning is dependent on the lower bound, it is crucial to be able to produce high-scoring alignments that satisfy the agreement constraints. Unfortunately, this problem is non-trivial. For instance, taking the union of directional alignments does not guarantee a feasible solution; whereas taking the intersection is trivially feasible but often not high-scoring.

To produce higher-scoring feasible bidirectional alignments we introduce a greedy heuristic algorithm. The algorithm starts with any feasible alignment (x, y, z). It runs the following greedy loop:

1. Repeat until there exists no $x(i, 0) = 1$ or $y(0, j) = 1$, or there is no score increase.
   (a) For each $i \in [I], j \in [J]_0, k \in \mathcal{K}$: $x(i, 0) = 1$, check if $x(i, j) \leftarrow 1$ and $y(i, j + k) \leftarrow 1$ is feasible, remember score.
   (b) For each $i \in [I]_0, j \in [J], k \in \mathcal{K}$: $y(0, j) = 1$, check if $y(i, j) \leftarrow 1$ and $x(i + k, j) \leftarrow 1$ is feasible, remember score.
   (c) Let (x, y, z) be the highest-scoring feasible solution produced.

This algorithm produces feasible alignments with monotonically increasing score, starting from the intersection of the alignments. It has run-time of $O(IJ(I + J))$ since each inner loop enumerates IJ possible updates and assigns at least one index a non-zero value, limiting the outer loop to $I + J$ iterations.

In practice we initialize the heuristic based on the intersection of x and y at the current round of Lagrangian relaxation. Experiments show that running this algorithm significantly improves the lower bound compared to just taking the intersection, and consequently helps pruning significantly.

7 Related Work

The most common techniques for bidirectional alignment are post-hoc combinations, such as union or intersection, of directional models, (Och et al., 1999), or more complex heuristic combiners such as grow-diag-final (Koehn et al., 2003).

Several authors have explored explicit bidirectional models in the literature. Cromieres and Kurohashi (2009) use belief propagation on a factor graph to train and decode a one-to-one word alignment problem. Qualitatively this method is similar to ours, although the model and decoding algorithm are different, and their method is not able to provide certificates of optimality.

A series of papers by Ganchev et al. (2010), Graca et al. (2008), and Ganchev et al. (2008) use posterior regularization to constrain the posterior probability of the word alignment problem to be symmetric and bijective. This work acheives state-of-the-art performance for alignment. Instead of utilizing posteriors our model tries to decode a single best one-to-one word alignment.

A different approach is to use constraints at training time to obtain models that favor bidirectional properties. Liang et al. (2006) propose agreement-based learning, which jointly learns probabilities by maximizing a combination of likelihood and agreement between two directional models.

General linear programming approaches have also been applied to word alignment problems. Lacoste-Julien et al. (2006) formulate the word alignment problem as quadratic assignment problem and solve it using an integer linear programming solver.

Our work is most similar to DeNero and Macherey (2011), which uses dual decomposition to encourage agreement between two directional HMM aligners during decoding time.

8 Experiments

Our experimental results compare the accuracy and optimality of our decoding algorithm to directional alignment models and previous work on this bidirectional model.

Data and Setup The experimental setup is identical to DeNero and Macherey (2011). Evaluation is performed on a hand-aligned subset of the NIST 2002 Chinese-English dataset (Ayan and Dorr, 2006). Following past work, the first 150 sentence pairs of the training section are used for evaluation. The potential parameters θ and ω are set based on unsupervised HMM models trained on the LDC FBIS corpus (6.2 million words).

	1-20 (28%)			21-40 (45%)			41-60 (27%)			all		
	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact
ILP	15.12	100.0	100.0	364.94	100.0	100.0	2,829.64	100.0	100.0	924.24	100.0	100.0
LR	0.55	97.6	97.6	4.76	55.9	55.9	15.06	7.5	7.5	6.33	54.7	54.7
CONS	0.43	100.0	100.0	9.86	95.6	95.6	61.86	55.0	62.5	21.08	86.0	88.0
D&M	-	6.2	-	-	0.0	-	-	0.0	-	-	6.2	-

Table 1: Experimental results for model accuracy of bilingual alignment. Column *time* is the mean time per sentence pair in seconds; *cert* is the percentage of sentence pairs solved with a certificate of optimality; *exact* is the percentage of sentence pairs solved exactly. Results are grouped by sentence length. The percentage of sentence pairs in each group is shown in parentheses.

Training is performed using the agreement-based learning method which encourages the directional models to overlap (Liang et al., 2006). This directional model has been shown produce state-of-the-art results with this setup (Haghighi et al., 2009).

Baselines We compare the algorithm described in this paper with several baseline methods. DIR includes post-hoc combinations of the $\mathbf{e}\rightarrow\mathbf{f}$ and $\mathbf{f}\rightarrow\mathbf{e}$ HMM-based aligners. Variants include union, intersection, and grow-diag-final. D&M is the dual decomposition algorithm for bidirectional alignment as presented by DeNero and Macherey (2011) with different final combinations. LR is the Lagrangian relaxation algorithm applied to the adjacent agreement problem without the additional constraints described in Section 5. CONS is our full Lagrangian relaxation algorithm including incremental constraint addition. ILP uses a highly-optimized general-purpose integer linear programming solver to solve the lattice with the constraints described (Gurobi Optimization, 2013).

Implementation The main task of the decoder is to repeatedly compute the $\arg\max$ of $L(\lambda)$. To speed up decoding, our implementation fully instantiates the Viterbi lattice for a problem instance. This approach has several benefits: each iteration can reuse the same lattice structure; max-marginals can be easily computed with a general forward-backward algorithm; pruning corresponds to removing lattice edges; and adding constraints can be done through lattice intersection. For consistency, we implement each baseline (except for D&M) through the same lattice.

Parameter Settings We run 400 iterations of the subgradient algorithm using the rate schedule $\eta_t = 0.95^{t'}$ where t' is the count of updates for which the dual value did not improve. Every 10 iterations we run the greedy decoder to compute a lower bound. If the gap between our current dual value $L(\lambda)$ and the lower bound improves significantly we run coarse-to-fine pruning as described in Section 6 with the best lower bound. For CONS, if the algorithm does not find an optimal solution we run 400 more iterations and incrementally add the 5 most violated constraints every 25 iterations.

Model	Combiner	alignment			phrase pair		
		Prec	Rec	AER	Prec	Rec	F1
DIR	union	57.6	80.0	33.4	75.1	33.5	46.3
	intersection	86.2	62.9	27.0	64.3	43.5	51.9
	grow-diag	59.7	79.5	32.1	70.1	36.9	48.4
D&M	union	63.3	81.5	29.1	63.2	44.9	52.5
	intersection	77.5	75.1	23.6	57.1	53.6	55.3
	grow-diag	65.6	80.6	28.0	60.2	47.4	53.0
CONS		72.5	74.9	26.4	53.0	52.4	52.7

Table 2: Alignment accuracy and phrase pair extraction accuracy for directional and bidirectional models. `Prec` is the precision. `Rec` is the recall. `AER` is alignment error rate and `F1` is the phrase pair extraction F1 score.

Results Our first set of experiments looks at the model accuracy and the decoding time of various methods that can produce optimal solutions. Results are shown in Table 1. D&M is only able to find the optimal solution with certificate on 6% of instances. The relaxation algorithm used in this work is able to increase that number to 54.7%. With incremental constraints and pruning, we are able to solve over 86% of sentence pairs including many longer and more difficult pairs. Additionally the method finds these solutions with only a small increase in running time over Lagrangian relaxation, and is significantly faster than using an ILP solver.

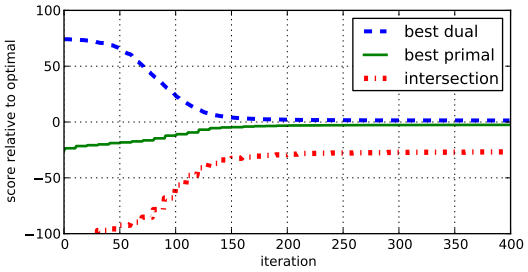
Next we compare the models in terms of alignment accuracy. Table 2 shows the precision, recall and alignment error rate (AER) for word alignment. We consider union, intersection and grow-diag-final as combination procedures. The combination procedures are applied to D&M in the case when the algorithm does not converge. For CONS, we use the optimal solution for the 86% of instances that converge and the highest-scoring greedy solution for those that do not. The proposed method has an AER of 26.4, which outperforms each of the directional models. However, although CONS achieves a higher model score than D&M, it performs worse in accuracy. Ta-

	1-20	21-40	41-60	all
# cons.	20.0	32.1	39.5	35.9

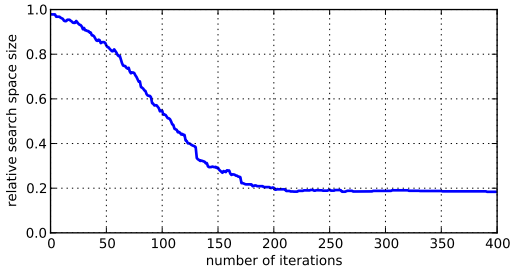
Table 3: The average number of constraints added for sentence pairs where Lagrangian relaxation is not able to find an exact solution.

ble 2 also compares the models in terms of phrase-extraction accuracy (Ayan and Dorr, 2006). We use the phrase extraction algorithm described by DeNero and Klein (2010), accounting for possible links and ϵ alignments. CONS performs better than each of the directional models, but worse than the best D&M model.

Finally we consider the impact of constraint addition, pruning, and use of a lower bound. Table 3 gives the average number of constraints added for sentence pairs for which Lagrangian relaxation alone does not produce a certificate. Figure 7(a) shows the average over all sentence pairs of the best dual and best primal scores. The graph compares the use of the greedy algorithm from Section 6.2 with the simple intersection of x and y. The difference between these curves illustrates the benefit of the greedy algorithm. This is reflected in Figure 7(b) which shows the effectiveness of coarse-to-fine pruning over time. On average, the pruning reduces the search space of each sentence pair to 20% of the initial search space after 200 iterations.



(a) The best dual and the best primal score, relative to the optimal score, averaged over all sentence pairs. The `best primal` curve uses a feasible greedy algorithm, whereas the `intersection` curve is calculated by taking the intersection of x and y.



(b) A graph showing the effectiveness of coarse-to-fine pruning. Relative search space size is the size of the pruned lattice compared to the initial size. The plot shows an average over all sentence pairs.

Figure 7

9 Conclusion

We have introduced a novel Lagrangian relaxation algorithm for a bidirectional alignment model that uses incremental constraint addition and coarse-to-fine pruning to find exact solutions. The algorithm increases the number of exact solution found on the model of DeNero and Macherey (2011) from 6% to 86%.

Unfortunately despite achieving higher model score, this approach does not produce more accurate alignments than the previous algorithm. This suggests that the adjacent agreement model may still be too constrained for this underlying task. Implicitly, an approach with fewer exact solutions may allow for useful violations of these constraints. In future work, we hope to explore bidirectional models with soft-penalties to explicitly permit these violations.

A Proof of NP-Hardness

We can show that the bidirectional alignment problem is NP-hard by reduction from the traveling salesman problem (TSP). A TSP instance with N cities has distance $c(i', i)$ for each $(i', i) \in [N]^2$. We can construct a sentence pair in which $I = J = N$ and ϵ-alignments have infinite cost.

$$\begin{aligned}
\omega(i', i, j) &= -c(i', i) && \forall i' \in [N]_0, i \in [N], j \in [N] \\
\theta(j', i, j) &= 0 && \forall j' \in [N]_0, i \in [N], j \in [N] \\
\omega(i', 0, j) &= -\infty && \forall i' \in [N]_0, j \in [N] \\
\theta(j', i, 0) &= -\infty && \forall j' \in [N]_0, i \in [N]
\end{aligned}$$

Every bidirectional alignment with finite objective score must align exactly one word in e to each word in f, encoding a permutation a. Moreover, each possible permutation has a finite score: the negation of the total distance to traverse the N cities in order a under distance c. Therefore, solving such a bidirectional alignment problem would find a minimal Hamiltonian path of the TSP encoded in this way, concluding the reduction.

Acknowledgments Alexander Rush, Yin-Wen Chang and Michael Collins were all supported by NSF grant IIS-1161814. Alexander Rush was partially supported by an NSF Graduate Research Fellowship.

References

Necip Fazil Ayan and Bonnie J Dorr. 2006. Going beyond aer: An extensive analysis of word alignments and their impact on mt. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 9–16. Association for Computational Linguistics.

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

Fabien Cromieres and Sadao Kurohashi. 2009. An alignment algorithm using belief propagation and a structure-based distortion model. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 166–174. Association for Computational Linguistics.

John DeNero and Dan Klein. 2010. Discriminative modeling of extraction sets for machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1453–1463. Association for Computational Linguistics.

John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *ACL*, pages 420–429.

Kuzman Ganchev, João V. Graça, and Ben Taskar. 2008. Better alignments = better translations? In *Proceedings of ACL-08: HLT*, pages 986–993, Columbus, Ohio, June. Association for Computational Linguistics.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning Research*, 11:2001–2049.

Joao Graca, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press, Cambridge, MA.

Inc. Gurobi Optimization. 2013. Gurobi optimizer reference manual.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 923–931. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I Jordan. 2006. Word alignment via quadratic assignment. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 112–119. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Robert C Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 81–88. Association for Computational Linguistics.

Franz Josef Och, Christoph Tillmann, Hermann Ney, et al. 1999. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Alexander M Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.