Lagrangian Relaxation Algorithms for Inference in Natural Language Processing

Alexander M. Rush and Michael Collins

(based on joint work with Yin-Wen Chang, Tommi Jaakkola, Terry Koo, Roi Reichart, David Sontag)
Decoding in NLP

**focus:** structured prediction for natural language processing

decoding as a combinatorial optimization problem

\[ y^* = \arg \max_{y \in \mathcal{Y}} f(y) \]

where \( f \) is a scoring function and \( \mathcal{Y} \) is a set of structures

for some problems, use simple combinatorial algorithms

- dynamic programming
- minimum spanning tree
- min cut
Structured prediction: Tagging

United  flies  some  large  jet

N → V → D → A → N

United$_1$  flies$_2$  some$_3$  large$_4$  jet$_5$
Structured prediction: Parsing

United flies some large jet

*0 United1 flies2 some3 large4 jet5
Decoding complexity

**issue:** simple combinatorial algorithms do not scale to richer models

\[ y^* = \arg \max_{y \in \mathcal{Y}} f(y) \]

need decoding algorithms for complex natural language tasks

**motivation:**
- richer model structure often leads to improved accuracy
- exact decoding for complex models tends to be intractable
Structured prediction: Phrase-based translation

das muss unsere sorge gleichermaßen sein

das muss unsere sorge gleichermaßen sein
this must our concern also be our concern
Structured prediction: Word alignment

the ugly dog has red fur

le chien laid a fourrure rouge
Decoding tasks

high complexity
- combined parsing and part-of-speech tagging (Rush et al., 2010)
- “loopy” HMM part-of-speech tagging
- syntactic machine translation (Rush and Collins, 2011)

NP-Hard
- symmetric HMM alignment (DeNero and Macherey, 2011)
- phrase-based translation (Chang and Collins, 2011)
- higher-order non-projective dependency parsing (Koo et al., 2010)

in practice:
- approximate decoding methods (coarse-to-fine, beam search, cube pruning, gibbs sampling, belief propagation)
- approximate models (mean field, variational models)
Lagrangian relaxation

a general technique for constructing decoding algorithms

solve complicated models

\[ y^* = \arg \max_y f(y) \]

by decomposing into smaller problems.

**upshot:** can utilize a toolbox of combinatorial algorithms.

- dynamic programming
- minimum spanning tree
- shortest path
- min cut
- ...
Lagrangian relaxation algorithms

Simple - uses basic combinatorial algorithms
Efficient - faster than solving exact decoding problems

Strong guarantees
- gives a certificate of optimality when exact
- direct connections to linear programming relaxations
MAP problem in Markov random fields

given: binary variables $x_1 \ldots x_n$

goal: MAP problem

$$\arg \max_{x_1 \ldots x_n} \sum_{(i,j) \in E} f_{i,j}(x_i, x_j)$$

where each $f_{i,j}(x_i, x_j)$ is a local potential for variables $x_i, x_j$
Dual decomposition for MRFs (Komodakis et al., 2010)

Goal:
\[
\arg \max_{x_1 \ldots x_n} \sum_{(i,j) \in E} f_{i,j}(x_i, x_j)
\]

Equivalent formulation:
\[
\arg \max_{x_1 \ldots x_n, y_1 \ldots y_n} \sum_{(i,j) \in T_1} f'_{i,j}(x_i, x_j) + \sum_{(i,j) \in T_2} f'_{i,j}(y_i, y_j)
\]

such that for \(i = 1 \ldots n\),
\[
x_i = y_i
\]

Lagrangian:
\[
L(u, x, y) = \sum_{(i,j) \in T_1} f'_{i,j}(x_i, x_j) + \sum_{(i,j) \in T_2} f'_{i,j}(y_i, y_j) + \sum_i u_i(x_i - y_i)
\]
Related work

- belief propagation using combinatorial algorithms (Duchi et al., 2007; Smith and Eisner, 2008)
- factored A* search (Klein and Manning, 2003)
Tutorial outline

1. worked algorithm for combined parsing and tagging
2. important theorems and formal derivation
3. more examples from parsing and alignment
4. relationship to linear programming relaxations
5. practical considerations for implementation
6. further example from machine translation
1. Worked example

aim: walk through a Lagrangian relaxation algorithm for combined parsing and part-of-speech tagging

• introduce formal notation for parsing and tagging
• give assumptions necessary for decoding
• step through a run of the Lagrangian relaxation algorithm
Combined parsing and part-of-speech tagging

**goal:** find parse tree that optimizes

\[ score(S \rightarrow NP \ VP) + score(VP \rightarrow V \ NP) + \]

\[ ... + score(N \rightarrow V) + score(N \rightarrow United) + ... \]
Constituency parsing

notation:
- $\mathcal{Y}$ is set of constituency parses for input
- $y \in \mathcal{Y}$ is a valid parse
- $f(y)$ scores a parse tree

goal:

$$\arg \max_{y \in \mathcal{Y}} f(y)$$

diagram example: a context-free grammar for constituency parsing
Part-of-speech tagging

notation:

- $\mathcal{Z}$ is set of tag sequences for input
- $z \in \mathcal{Z}$ is a valid tag sequence
- $g(z)$ scores of a tag sequence

goal:

$$\arg\max_{z \in \mathcal{Z}} g(z)$$

example: an HMM for part-of-speech tagging

```
N  V  D  A  N
  ↓  ↓  ↓  ↓  ↓
United, flies, some, large, jet
```
Identifying tags

**notation:** identify the tag labels selected by each model

- $y(i, t) = 1$ when parse $y$ selects tag $t$ at position $i$
- $z(i, t) = 1$ when tag sequence $z$ selects tag $t$ at position $i$

**example:** a parse and tagging with $y(4, A) = 1$ and $z(4, A) = 1$

```
S
  NP
    N United
    V flies
  VP
    NP
      D A N
        some large jet
Y

N → V → D → A → N
  ↓  ↓  ↓  ↓  ↓
United₁ flies₂ some₃ large₄ jet₅
```

$Z$
Combined optimization

goal:
\[ \arg \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} f(y) + g(z) \]
such that for all \( i = 1 \ldots n, t \in T \),
\[ y(i, t) = z(i, t) \]
i.e. find the best parse and tagging pair that agree on tag labels

equivalent formulation:
\[ \arg \max_{y \in \mathcal{Y}} f(y) + g(l(y)) \]
where \( l : \mathcal{Y} \to \mathcal{Z} \) extracts the tag sequence from a parse tree
Exact method: Dynamic programming intersection can solve by solving the product of the two models

example:

- parsing model is a context-free grammar
- tagging model is a first-order HMM
- can solve as CFG and finite-state automata intersection

replace $VP \rightarrow V \ NP$
with $VP_{N,V} \rightarrow V_{N,V} \ NP_{V,N}$
Intersected parsing and tagging complexity

let $G$ be the number of grammar non-terminals

parsing CFG require $O(G^3 n^3)$ time with rules $VP \rightarrow V \ NP$

with intersection $O(G^3 n^3 |T|^3)$ with rules $VP_{N,V} \rightarrow V_{N,V} \ NP_{V,N}$

becomes $O(G^3 n^3 |T|^6)$ time for second-order HMM
Parsing assumption

**assumption:** optimization with $u$ can be solved efficiently

$$\arg \max_{y \in \mathcal{Y}} f(y) + \sum_{i,t} u(i, t)y(i, t)$$

**example:** CFG with rule scoring function $h$

$$f(y) = \sum_{X \rightarrow Y \ Z \in y} h(X \rightarrow Y \ Z) + \sum_{(i, X) \in y} h(X \rightarrow w_i)$$

where

$$\arg \max_{y \in \mathcal{Y}} f(y) + \sum_{i,t} u(i, t)y(i, t) =$$

$$\arg \max_{y \in \mathcal{Y}} \sum_{X \rightarrow Y \ Z \in y} h(X \rightarrow Y \ Z) + \sum_{(i, X) \in y} (h(X \rightarrow w_i) + u(i, X))$$
Tagging assumption

**assumption:** optimization with $u$ can be solved efficiently

$$\arg\max_{z \in \mathcal{Z}} g(z) - \sum_{i,t} u(i, t) z(i, t)$$

**example:** HMM with scores for transitions $T$ and observations $O$

$$g(z) = \sum_{t \rightarrow t' \in z} T(t \rightarrow t') + \sum_{(i,t) \in z} O(t \rightarrow w_i)$$

where

$$\arg\max_{z \in \mathcal{Z}} g(z) - \sum_{i,t} u(i, t) z(i, t) =$$

$$\arg\max_{z \in \mathcal{Z}} \sum_{t \rightarrow t' \in z} T(t \rightarrow t') + \sum_{(i,t) \in z} (O(t \rightarrow w_i) - u(i, t))$$
Lagrangian relaxation algorithm

Set $u^{(1)}(i, t) = 0$ for all $i, t \in \mathcal{T}$

For $k = 1$ to $K$

\[ y^{(k)} \leftarrow \arg \max_{y \in \mathcal{Y}} f(y) + \sum_{i, t} u^{(k)}(i, t)y(i, t) \] [Parsing]

\[ z^{(k)} \leftarrow \arg \max_{z \in \mathcal{Z}} g(z) - \sum_{i, t} u^{(k)}(i, t)z(i, t) \] [Tagging]

If $y^{(k)}(i, t) = z^{(k)}(i, t)$ for all $i, t$ Return $(y^{(k)}, z^{(k)})$

Else $u^{(k+1)}(i, t) \leftarrow u^{(k)}(i, t) - \alpha_k(y^{(k)}(i, t) - z^{(k)}(i, t))$
CKY Parsing

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t)y(i, t)) \]

Viterbi Decoding

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t)z(i, t)) \]

Key

\[ f(y) \leftarrow \text{CFG} \]
\[ g(z) \leftarrow \text{HMM} \]
\[ \mathcal{Y} \leftarrow \text{Parse Trees} \]
\[ \mathcal{Z} \leftarrow \text{Taggings} \]

\[ y(i, t) = 1 \text{ if } y \text{ contains tag } t \text{ at position } i \]
CKY Parsing

\[ S \]
\[ NP \]
\[ A \]
\[ United \]
\[ N \]
\[ flies \]
\[ D \]
\[ some \]
\[ A \]
\[ large \]
\[ VP \]
\[ V \]
\[ jet \]

\[ y^* = \arg \max_{y \in Y} (f(y) + \sum_{i,t} u(i, t)y(i, t)) \]

Viterbi Decoding

United_1 flies_2 some_3 large_4 jet_5

\[ z^* = \arg \max_{z \in Z} (g(z) - \sum_{i,t} u(i, t)z(i, t)) \]

Key

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(y) )</td>
<td>( \Leftarrow )</td>
<td>CFG</td>
</tr>
<tr>
<td>( g(z) )</td>
<td>( \Leftarrow )</td>
<td>HMM</td>
</tr>
<tr>
<td>( Y )</td>
<td>( \Leftarrow )</td>
<td>Parse Trees</td>
</tr>
<tr>
<td>( Z )</td>
<td>( \Leftarrow )</td>
<td>Taggings</td>
</tr>
<tr>
<td>( y(i, t) = 1 ) if ( y ) contains tag ( t ) at position ( i )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Penalties

\[ u(i, t) = 0 \text{ for all } i, t \]
CKY Parsing

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t)y(i, t)) \]

Viterbi Decoding

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t)z(i, t)) \]

Key

\[ f(y) \leftarrow \text{CFG} \quad g(z) \leftarrow \text{HMM} \]
\[ \mathcal{Y} \leftarrow \text{Parse Trees} \quad \mathcal{Z} \leftarrow \text{Taggings} \]
\[ y(i, t) = 1 \quad \text{if} \quad y \text{ contains tag } t \text{ at position } i \]
CKY Parsing

$$y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t)y(i, t))$$

Viterbi Decoding

$$z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t)z(i, t))$$

Key

- \(f(y) \leftarrow \text{CFG}\)
- \(g(z) \leftarrow \text{HMM}\)
- \(\mathcal{Y} \leftarrow \text{Parse Trees}\)
- \(\mathcal{Z} \leftarrow \text{Taggings}\)
- \(y(i, t) = 1\) if \(y\) contains tag \(t\) at position \(i\)

Penalties

$$u(i, t) = 0 \text{ for all } i,t$$
**CKY Parsing**

$\begin{align*}
    S & \rightarrow NP \rightarrow VP \\
    NP & \rightarrow A \ N \\
    VP & \rightarrow A \ V \\
    A & \rightarrow \text{United} \\
    N & \rightarrow \text{flies} \\
    D & \rightarrow \text{some} \\
    A & \rightarrow \text{large} \\
    V & \rightarrow \text{jet}
\end{align*}$

$y^* = \arg \max_{y \in Y} (f(y) + \sum_{i,t} u(i, t) y(i, t))$

**Viterbi Decoding**

$\begin{align*}
    N \rightarrow V \rightarrow D \rightarrow A \rightarrow N \\
    \downarrow \downarrow \downarrow \downarrow \downarrow \\
    \text{United}_1 \text{flies}_2 \text{some}_3 \text{large}_4 \text{jet}_5
\end{align*}$

$z^* = \arg \max_{z \in Z} (g(z) - \sum_{i,t} u(i, t) z(i, t))$

**Key**

- $f(y) \Leftarrow \text{CFG}$
- $g(z) \Leftarrow \text{HMM}$
- $Y \Leftarrow \text{Parse Trees}$
- $Z \Leftarrow \text{Taggings}$
- $y(i, t) = 1$ if $y$ contains tag $t$ at position $i$

**Penalties**

$u(i, t) = 0$ for all $i, t$

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$u(1, A)$</td>
<td>-1</td>
</tr>
<tr>
<td>$u(1, N)$</td>
<td>1</td>
</tr>
<tr>
<td>$u(2, N)$</td>
<td>-1</td>
</tr>
<tr>
<td>$u(2, V)$</td>
<td>1</td>
</tr>
<tr>
<td>$u(5, V)$</td>
<td>-1</td>
</tr>
<tr>
<td>$u(5, N)$</td>
<td>1</td>
</tr>
</tbody>
</table>
CKY Parsing

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i, t} u(i, t)y(i, t)) \]

Viterbi Decoding

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i, t} u(i, t)z(i, t)) \]

Key

\[ f(y) \leftarrow \text{CFG} \quad g(z) \leftarrow \text{HMM} \]
\[ \mathcal{Y} \leftarrow \text{Parse Trees} \quad \mathcal{Z} \leftarrow \text{Taggings} \]
\[ y(i, t) = 1 \quad \text{if} \quad y \text{ contains tag } t \text{ at position } i \]

Penalties

\[ u(i, t) = 0 \text{ for all } i, t \]

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( u(1, A) )</td>
<td>-1</td>
</tr>
<tr>
<td>( u(1, N) )</td>
<td>1</td>
</tr>
<tr>
<td>( u(2, N) )</td>
<td>-1</td>
</tr>
<tr>
<td>( u(2, V) )</td>
<td>1</td>
</tr>
<tr>
<td>( u(5, V) )</td>
<td>-1</td>
</tr>
<tr>
<td>( u(5, N) )</td>
<td>1</td>
</tr>
</tbody>
</table>
CKY Parsing

\[ y^* = \arg \max_{y \in Y} (f(y) + \sum_{i,t} u(i, t)y(i, t)) \]

Viterbi Decoding

\[ z^* = \arg \max_{z \in Z} (g(z) - \sum_{i,t} u(i, t)z(i, t)) \]

Key

- \( f(y) \) \leftarrow CFG
- \( g(z) \) \leftarrow HMM
- \( Y \) \leftarrow Parse Trees
- \( Z \) \leftarrow Taggings
- \( y(i, t) = 1 \) if \( y \) contains tag \( t \) at position \( i \)

Penalties

\[ u(i, t) = 0 \text{ for all } i, t \]

Iteration 1

\begin{align*}
\text{u(1, A)} & \quad -1 \\
\text{u(1, N)} & \quad 1 \\
\text{u(2, N)} & \quad -1 \\
\text{u(2, V)} & \quad 1 \\
\text{u(5, V)} & \quad -1 \\
\text{u(5, N)} & \quad 1
\end{align*}

Converged

\[ y^* = \arg \max_{y \in Y} f(y) + g(y) \]
CKY Parsing

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t)y(i, t)) \]

Viterbi Decoding

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t)z(i, t)) \]

Penalties

\[ u(i, t) = 0 \text{ for all } i,t \]

Iteration 1

\[
\begin{array}{c|c}
  & -1 \\
 1 & \text{A} \\
 1 & \text{N} \\
 2 & \text{D} \\
 2 & \text{A} \\
 2 & \text{N} \\
 5 & \text{V} \\
 5 & \text{N} \\
 5 & \text{V} \\
 5 & \text{N} \\
\end{array}
\]

Converged

\[ y^* = \arg \max_{y \in \mathcal{Y}} f(y) + g(z) \]

Key

\[
\begin{align*}
  f(y) & \Leftarrow \text{CFG} \\
  \mathcal{Y} & \Leftarrow \text{Parse Trees} \\
  y(i, t) = 1 & \text{ if } y \text{ contains tag } t \text{ at position } i
\end{align*}
\]
**CKY Parsing**

\[
y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t)y(i, t))
\]

**Viterbi Decoding**

\[
z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t)z(i, t))
\]

**Key**

\[
\begin{align*}
f(y) & \leftarrow \text{CFG} \\
\mathcal{Y} & \leftarrow \text{Parse Trees} \\
y(i, t) = 1 & \text{ if } y \text{ contains tag } t \text{ at position } i
\end{align*}
\]

**Penalties**

\[
u(i, t) = 0 \text{ for all } i, t
\]

**Iteration 1**

\[
\begin{array}{c|c}
\text{u(1, A)} & -1 \\
\text{u(1, N)} & 1 \\
\text{u(2, N)} & -1 \\
\text{u(2, V)} & 1 \\
\text{u(5, V)} & -1 \\
\text{u(5, N)} & 1 \\
\end{array}
\]

**Iteration 2**

\[
\begin{array}{c|c}
\text{u(5, V)} & -1 \\
\text{u(5, N)} & 1 \\
\end{array}
\]
CKY Parsing

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t)y(i, t)) \]

Viterbi Decoding

United_1 flies_2 some_3 large_4 jet_5

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t)z(i, t)) \]

Key

\begin{align*}
f(y) & \leftarrow \text{CFG} \\
\mathcal{Y} & \leftarrow \text{Parse Trees} \\
y(i, t) = 1 & \text{ if } y \text{ contains tag } t \text{ at position } i
\end{align*}

Penalties

\[ u(i, t) = 0 \text{ for all } i,t \]

\begin{tabular}{l|l}
<table>
<thead>
<tr>
<th>Iteration 1</th>
<th>Iteration 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u(1, A) )</td>
<td>( u(1, A) )</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>( u(1, N) )</td>
<td>( u(1, N) )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( u(2, N) )</td>
<td>( u(2, N) )</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>( u(2, V) )</td>
<td>( u(2, V) )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( u(5, V) )</td>
<td>( u(5, V) )</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>( u(5, N) )</td>
<td>( u(5, N) )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
\end{tabular}
**CKY Parsing**

\[
y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t)y(i, t))
\]

**Viterbi Decoding**

United₁ flies₂ some₃ large₄ jet₅

\[
z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t)z(i, t))
\]

**Key**

\[
\begin{align*}
    f(y) & \leftarrow \text{CFG} \\
    \mathcal{Y} & \leftarrow \text{Parse Trees} \\
    y(i, t) = 1 & \text{ if } y \text{ contains tag } t \text{ at position } i \\
    g(z) & \leftarrow \text{HMM} \\
    \mathcal{Z} & \leftarrow \text{Taggings}
\end{align*}
\]

**Penalties**

\[
u(i, t) = 0 \text{ for all } i, t
\]

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th>Iteration 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>u(1, A)</td>
<td>u(5, V)</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>u(1, N)</td>
<td>u(5, N)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>u(2, N)</td>
<td>u(2, V)</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>u(2, V)</td>
<td>u(5, V)</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>u(5, N)</td>
<td>u(5, N)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
CKY Parsing

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t)y(i, t)) \]

Viterbi Decoding

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t)z(i, t)) \]

Key

\[
\begin{align*}
f(y) & \quad \Leftarrow \quad \text{CFG} \\
\mathcal{Y} & \quad \Leftarrow \quad \text{Parse Trees} \\
y(i, t) &= 1 \quad \text{if} \quad y \text{ contains tag } t \text{ at position } i
\end{align*}
\]

Penalties

\[
\begin{array}{c|c}
\text{Iteration 1} & \\
\hline
u(1, A) & -1 \\
u(1, N) & 1 \\
u(2, N) & -1 \\
u(2, V) & 1 \\
u(5, V) & -1 \\
u(5, N) & 1 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{Iteration 2} & \\
\hline
u(5, V) & -1 \\
u(5, N) & 1 \\
\end{array}
\]

\[ u(i, t) = 0 \text{ for all } i, t \]
CKY Parsing

\[
y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i, t)y(i, t))
\]

Viterbi Decoding

\[
z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i, t)z(i, t))
\]

Key

\[
\begin{align*}
f(y) & \Leftarrow \text{CFG} \\
\mathcal{Y} & \Leftarrow \text{Parse Trees} \\
y(i, t) = 1 & \text{ if } y \text{ contains tag } t \text{ at position } i \\
g(z) & \Leftarrow \text{HMM} \\
\mathcal{Z} & \Leftarrow \text{Taggings}
\end{align*}
\]

Penalties

\[
u(i, t) = 0 \text{ for all } i,t
\]

Iteration 1

\[
\begin{array}{l|l}
\hline
& \\ 
u(1, A) & -1 \\
u(1, N) & 1 \\
u(2, N) & -1 \\
u(2, V) & 1 \\
u(5, V) & -1 \\
u(5, N) & 1 \\
\hline
\end{array}
\]

Iteration 2

\[
\begin{array}{l|l}
\hline
& \\ 
u(5, V) & -1 \\
u(5, N) & 1 \\
\hline
\end{array}
\]

Converged

\[
y^* = \arg \max_{y \in \mathcal{Y}} f(y) + g(y)
\]
Main theorem

**Theorem:** if at any iteration, for all \( i, t \in \mathcal{T} \)

\[
y^{(k)}(i, t) = z^{(k)}(i, t)
\]

then \((y^{(k)}, z^{(k)})\) is the global optimum

**Proof:** focus of the next section
Convergence

% examples converged vs. number of iterations
2. Formal properties

**aim**: formal derivation of the algorithm given in the previous section

- derive Lagrangian dual
- prove three properties
  - upper bound
  - convergence
  - optimality
- describe subgradient method
Lagrangian

goal:

\[ \arg \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} f(y) + g(z) \text{ such that } y(i, t) = z(i, t) \]

Lagrangian:

\[ L(u, y, z) = f(y) + g(z) + \sum_{i,t} u(i, t) (y(i, t) - z(i, t)) \]

redistribute terms

\[
L(u, y, z) = \left( f(y) + \sum_{i,t} u(i, t) y(i, t) \right) + \left( g(z) - \sum_{i,t} u(i, t) z(i, t) \right)
\]
Lagrangian dual

Lagrangian:

\[ L(u, y, z) = \left( f(y) + \sum_{i,t} u(i, t)y(i, t) \right) + \left( g(z) - \sum_{i,t} u(i, t)z(i, t) \right) \]

Lagrangian dual:

\[ L(u) = \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} L(u, y, z) \]

\[ = \max_{y \in \mathcal{Y}} \left( f(y) + \sum_{i,t} u(i, t)y(i, t) \right) + \max_{z \in \mathcal{Z}} \left( g(z) - \sum_{i,t} u(i, t)z(i, t) \right) \]
Theorem 1. Upper bound

define:

- \( y^*, z^* \) is the optimal combined parsing and tagging solution with \( y^*(i, t) = z^*(i, t) \) for all \( i, t \)

**Theorem:** for any value of \( u \)

\[
L(u) \geq f(y^*) + g(z^*)
\]

\( L(u) \) provides an upper bound on the score of the optimal solution

**Note:** upper bound may be useful as input to branch and bound or A* search
Theorem 1. Upper bound (proof)

**Theorem:** for any value of $u$, $L(u) \geq f(y^*) + g(z^*)$

**Proof:**

\[
L(u) = \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} L(u, y, z) \quad (1)
\]

\[
\geq \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} L(u, y, z) \quad (2)
\]

\[
= \max_{y \in \mathcal{Y}, z \in \mathcal{Z}: y = z} f(y) + g(z) \quad (3)
\]

\[
= f(y^*) + g(z^*) \quad (4)
\]
Formal algorithm (reminder)

Set $u^{(1)}(i, t) = 0$ for all $i, t \in \mathcal{T}$

For $k = 1$ to $K$

$$y^{(k)} \leftarrow \arg \max_{y \in \mathcal{Y}} f(y) + \sum_{i,t} u^{(k)}(i, t)y(i, t) \text{ [Parsing]}$$

$$z^{(k)} \leftarrow \arg \max_{z \in \mathcal{Z}} g(z) - \sum_{i,t} u^{(k)}(i, t)z(i, t) \text{ [Tagging]}$$

If $y^{(k)}(i, t) = z^{(k)}(i, t)$ for all $i, t$ Return $(y^{(k)}, z^{(k)})$

Else $u^{(k+1)}(i, t) \leftarrow u^{(k)}(i, t) - \alpha_k(y^{(k)}(i, t) - z^{(k)}(i, t))$
Theorem 2. Convergence

notation:
- $u^{(k+1)}(i, t) \leftarrow u^{(k)}(i, t) + \alpha_k(y^{(k)}(i, t) - z^{(k)}(i, t))$ is update
- $u^{(k)}$ is the penalty vector at iteration $k$
- $\alpha_k > 0$ is the update rate at iteration $k$

theorem: for any sequence $\alpha^1, \alpha^2, \alpha^3, \ldots$ such that

$$\lim_{t \to \infty} \alpha^t = 0 \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha^t = \infty,$$

we have

$$\lim_{t \to \infty} L(u^t) = \min_u L(u)$$

i.e. the algorithm converges to the tightest possible upper bound

proof: by subgradient convergence (next section)
Dual solutions

define:

• for any value of $u$

\[ y_u = \arg \max_{y \in \mathcal{Y}} \left( f(y) + \sum_{i,t} u(i, t)y(i, t) \right) \]

and

\[ z_u = \arg \max_{z \in \mathcal{Z}} \left( g(z) - \sum_{i,t} u(i, t)z(i, t) \right) \]

• $y_u$ and $z_u$ are the dual solutions for a given $u$
Theorem 3. Optimality

**Theorem:** if there exists $u$ such that

$$y_u(i, t) = z_u(i, t)$$

for all $i, t$ then

$$f(y_u) + g(z_u) = f(y^*) + g(z^*)$$

i.e. if the dual solutions agree, we have an optimal solution

$$(y_u, z_u)$$
Theorem 3. Optimality (proof)

**Theorem:** if $u$ such that $y_u(i, t) = z_u(i, t)$ for all $i, t$ then

$$f(y_u) + g(z_u) = f(y^*) + g(z^*)$$

**Proof:** by the definitions of $y_u$ and $z_u$

$$L(u) = f(y_u) + g(z_u) + \sum_{i,t} u(i, t)(y_u(i, t) - z_u(i, t))$$

$$= f(y_u) + g(z_u)$$

since $L(u) \geq f(y^*) + g(z^*)$ for all values of $u$

$$f(y_u) + g(z_u) \geq f(y^*) + g(z^*)$$

but $y^*$ and $z^*$ are optimal

$$f(y_u) + g(z_u) \leq f(y^*) + g(z^*)$$
Dual optimization

Lagrangian dual:

\[ L(u) = \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} L(u, y, z) \]

\[ = \max_{y \in \mathcal{Y}} \left( f(y) + \sum_{i, t} u(i, t)y(i, t) \right) + \max_{z \in \mathcal{Z}} \left( g(z) - \sum_{i, t} u(i, t)z(i, t) \right) \]

**goal:** dual problem is to find the tightest upper bound

\[ \min_u L(u) \]
Dual subgradient

\[ L(u) = \max_{y \in Y} \left( f(y) + \sum_{i,t} u(i, t)y(i, t) \right) + \max_{z \in Z} \left( g(z) - \sum_{i,t} u(i, t)z(i, t) \right) \]

**properties:**

- \( L(u) \) is convex in \( u \) (no local minima)
- \( L(u) \) is not differentiable (because of max operator)

handle non-differentiability by using subgradient descent

**define:** a subgradient of \( L(u) \) at \( u \) is a vector \( g_u \) such that for all \( v \)

\[ L(v) \geq L(u) + g_u \cdot (v - u) \]
Subgradient algorithm

\[ L(u) = \max_{y \in Y} \left( f(y) + \sum_{i, t} u(i, t)y(i, t) \right) + \max_{z \in Z} \left( g(z) - \sum_{i, j} u(i, t)z(i, t) \right) \]

recall, \( y_u \) and \( z_u \) are the argmax’s of the two terms

subgradient:

\[ g_u(i, t) = y_u(i, t) - z_u(i, t) \]

subgradient descent: move along the subgradient

\[ u'(i, t) = u(i, t) - \alpha (y_u(i, t) - z_u(i, t)) \]

guaranteed to find a minimum with conditions given earlier for \( \alpha \)
3. More examples

aim: demonstrate similar algorithms that can be applied to other decoding applications

- context-free parsing combined with dependency parsing
- combined translation alignment
Combined constituency and dependency parsing
(Rush et al., 2010)

**setup:** assume separate models trained for constituency and dependency parsing

**problem:** find constituency parse that maximizes the sum of the two models

**example:**
- combine lexicalized CFG with second-order dependency parser
Lexicalized constituency parsing

notation:
• $\mathcal{Y}$ is set of lexicalized constituency parses for input
• $y \in \mathcal{Y}$ is a valid parse
• $f(y)$ scores a parse tree

goal:
$$\arg\max_{y \in \mathcal{Y}} f(y)$$

d-example: a lexicalized context-free grammar

```
S(flies)
  NP(United)  VP(flies)
    N  V  NP(jet)
    United flies D A N
      some  large  jet
```
define:

- $\mathcal{Z}$ is set of dependency parses for input
- $z \in \mathcal{Z}$ is a valid dependency parse
- $g(z)$ scores a dependency parse

example:
Identifying dependencies

**notation:** identify the dependencies selected by each model

- \( y(i, j) = 1 \) when word \( i \) modifies of word \( j \) in constituency parse \( y \)
- \( z(i, j) = 1 \) when word \( i \) modifies of word \( j \) in dependency parse \( z \)

**example:** a constituency and dependency parse with \( y(3, 5) = 1 \) and \( z(3, 5) = 1 \)
Combined optimization

goal:

\[
\arg \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} f(y) + g(z)
\]

such that for all \( i = 1 \ldots n, j = 0 \ldots n, \)

\[
y(i, j) = z(i, j)
\]
CKY Parsing

$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j))$

Dependency Parsing

$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j))$

Key

$f(y) \iff \text{CFG}$

$g(z) \iff \text{Dependency Model}$

$\mathcal{Y} \iff \text{Parse Trees}$

$\mathcal{Z} \iff \text{Dependency Trees}$

$y(i,j) = 1$ if $y$ contains dependency $i,j$

Penalties

$u(i,j) = 0$ for all $i,j$
CKY Parsing

\[ y^* = \operatorname{arg\,max}_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j)) \]

Dependency Parsing

\[ z^* = \operatorname{arg\,max}_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j)) \]

Key

\[
\begin{array}{ccc}
  f(y) & \Leftarrow & \text{CFG} \\
  \mathcal{Y} & \Leftarrow & \text{Parse Trees} \\
  y(i,j) = 1 & \text{if} & y \text{ contains dependency } i,j \\
  g(z) & \Leftarrow & \text{Dependency Model} \\
  \mathcal{Z} & \Leftarrow & \text{Dependency Trees}
\end{array}
\]
CKY Parsing

```
NP
  |  |  |  
  N  V  D  NP
  United  flies  some  A  N
  large  jet
```

\[
y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j))
\]

Penalties

\[
u(i,j) = 0 \text{ for all } i,j
\]

Dependency Parsing

```
*0  United  flies  some  large  jet
```

\[
z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j))
\]

Key

\[
f(y) \iff \text{CFG} \quad g(z) \iff \text{Dependency Model}
\]

\[
\mathcal{Y} \iff \text{Parse Trees} \quad \mathcal{Z} \iff \text{Dependency Trees}
\]

\[
y(i,j) = 1 \text{ if } y \text{ contains dependency } i,j
\]
CKY Parsing

\[ S (\text{flies}) \]  
\[ \text{NP} \rightarrow \text{N} \text{VP}(\text{flies}) \]  
\[ \text{NP(United)} \rightarrow \text{V D NP(jet)} \]  
\[ \text{V flies some A large jet} \]  
\[ \text{y}^* = \arg \max_{y \in Y} (f(y) + \sum_{i,j} u(i,j)y(i,j)) \]

Penalties

\[ u(i,j) = 0 \text{ for all } i,j \]

Dependency Parsing

\[ *_0 \rightarrow \text{United} \rightarrow \text{flies} \rightarrow \text{some} \rightarrow \text{large} \rightarrow \text{jet} \]

\[ z^* = \arg \max_{z \in Z} (g(z) - \sum_{i,j} u(i,j)z(i,j)) \]

Key

\[ f(y) \leftarrow \text{CFG} \]
\[ g(z) \leftarrow \text{Dependency Model} \]
\[ Y \leftarrow \text{Parse Trees} \]
\[ Z \leftarrow \text{Dependency Trees} \]
\[ y(i,j) = 1 \text{ if } y \text{ contains dependency } i,j \]
CKY Parsing

Dependency Parsing

Penalties

y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j))

z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j))

Key

f(y) \leftarrow \text{CFG} \quad g(z) \leftarrow \text{Dependency Model}

\mathcal{Y} \leftarrow \text{Parse Trees} \quad \mathcal{Z} \leftarrow \text{Dependency Trees}

y(i,j) = 1 \quad \text{if} \quad y \text{ contains dependency } i,j

u(i,j) = 0 \quad \text{for all } i,j

\begin{array}{c|c}
\text{Iteration 1} & \\
\frac{u(2,3)}{u(5,3)} & -1 \quad 1 \\
\end{array}
CKY Parsing

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j)) \]

Dependency Parsing

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j)) \]

Key

\[
\begin{array}{ll}
  f(y) & \iff \text{CFG} \\
  \mathcal{Y} & \iff \text{Parse Trees} \\
  y(i,j) = 1 & \text{if } y \text{ contains dependency } i,j \\
  g(z) & \iff \text{Dependency Model} \\
  \mathcal{Z} & \iff \text{Dependency Trees}
\end{array}
\]
CKY Parsing

\[
y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j))
\]

Dependency Parsing

\[
z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j))
\]

Penalties

\[
u(i,j) = 0 \text{ for all } i,j
\]

Iteration 1

\[
\begin{array}{c|c}
  u(2,3) & -1 \\
u(5,3) & 1 
\end{array}
\]

Key

\[
\begin{array}{ll}
f(y) & \Leftarrow \text{CFG} \\
\mathcal{Y} & \Leftarrow \text{Parse Trees} \\
y(i,j) = 1 & \text{if } y \text{ contains dependency } i,j \\
g(z) & \Leftarrow \text{Dependency Model} \\
\mathcal{Z} & \Leftarrow \text{Dependency Trees}
\end{array}
\]
CKY Parsing

\[
y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j))
\]

Dependency Parsing

\[
z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j))
\]

Key

\[
\begin{align*}
f(y) & \iff \text{CFG} \\
\mathcal{Y} & \iff \text{Parse Trees} \\
y(i,j) & = 1 \text{ if } y \text{ contains dependency } i,j \\
g(z) & \iff \text{Dependency Model} \\
\mathcal{Z} & \iff \text{Dependency Trees}
\end{align*}
\]

Penalties

\[
u(i,j) = 0 \text{ for all } i,j
\]

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(u(2,3))</td>
<td>-1</td>
</tr>
<tr>
<td>(u(5,3))</td>
<td>1</td>
</tr>
</tbody>
</table>
CKY Parsing

\[
y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j))
\]

Dependency Parsing

\[
z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j))
\]

Key

\[
\begin{align*}
  f(y) & \iff \text{CFG} \\
  \mathcal{Y} & \iff \text{Parse Trees} \\
  y(i,j) = 1 & \text{ if } y \text{ contains dependency } i,j \\
  g(z) & \iff \text{Dependency Model} \\
  \mathcal{Z} & \iff \text{Dependency Trees}
\end{align*}
\]

Penalties

\[
u(i,j) = 0 \text{ for all } i,j
\]

| Iteration 1 |  \\
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$u(2,3)$</td>
<td>-1</td>
</tr>
<tr>
<td>$u(5,3)$</td>
<td>1</td>
</tr>
</tbody>
</table>

Converged

\[
y^* = \arg \max_{y \in \mathcal{Y}} f(y) + g(y)
\]
Convergence

![Graph showing convergence with varying number of iterations.]
Integrated Constituency and Dependency Parsing: Accuracy

F₁ Score

- Collins (1997) Model 1
- Fixed, First-best Dependencies from Koo (2008)
- Dual Decomposition
setup: assume separate models trained for English-to-French and French-to-English alignment

problem: find an alignment that maximizes the score of both models

example:
- HMM models for both directional alignments (assume correct alignment is one-to-one for simplicity)
English-to-French alignment

define:

- \( \mathcal{Y} \) is set of all possible English-to-French alignments
- \( y \in \mathcal{Y} \) is a valid alignment
- \( f(y) \) scores of the alignment

example: HMM alignment

```
Le_1 \rightarrow laid_3 \rightarrow chien_2 \rightarrow a_4 \rightarrow rouge_6 \rightarrow fourrure_5

The_1 \downarrow ugly_2 \downarrow dog_3 \downarrow has_4 \downarrow red_5 \downarrow fur_6
```
French-to-English alignment

define:
- $\mathcal{Z}$ is set of all possible French-to-English alignments
- $z \in \mathcal{Z}$ is a valid alignment
- $g(z)$ scores of an alignment

example: HMM alignment

```
The_1    ugly_2    dog_3    has_4    fur_6    red_5
  ↓        ↓        ↓        ↓        ↓        ↓
Le_1  chien_2  laid_3  a_4  fourrure_5  rouge_6
```
Identifying word alignments

**notation:** identify the tag labels selected by each model

- \( y(i, j) = 1 \) when e-to-f alignment \( y \) selects French word \( i \) to align with English word \( j \)
- \( z(i, j) = 1 \) when f-to-e alignment \( z \) selects French word \( i \) to align with English word \( j \)

**example:** two HMM alignment models with \( y(6, 5) = 1 \) and \( z(6, 5) = 1 \)
Combined optimization

global optimization:

\[
\arg \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} f(y) + g(z)
\]

such that for all \( i = 1 \ldots n, \ j = 1 \ldots n, \)

\[
y(i, j) = z(i, j)
\]
English-to-French

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i, j)y(i, j)) \]

French-to-English

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i, j)z(i, j)) \]

Key

- \( f(y) \leftarrow \text{HMM Alignment} \)
- \( g(z) \leftarrow \text{HMM Alignment} \)
- \( \mathcal{Y} \leftarrow \text{English-to-French model} \)
- \( \mathcal{Z} \leftarrow \text{French-to-English model} \)
- \( y(i, j) = 1 \) if French word \( i \) aligns to English word \( j \)

Penalties

u(i, j) = 0 for all \( i, j \)
### English-to-French

$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j))$$

### French-to-English

$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j))$$

### Penalties

$$u(i,j) = 0 \text{ for all } i,j$$

### Key

- $f(y) \quad \iff \quad \text{HMM Alignment}$
- $g(z) \quad \iff \quad \text{HMM Alignment}$
- $\mathcal{Y} \quad \iff \quad \text{English-to-French model}$
- $\mathcal{Z} \quad \iff \quad \text{French-to-English model}$
- $y(i,j) = 1 \quad \text{if} \quad \text{French word } i \text{ aligns to English word } j$
English-to-French

\[ y^* = \arg \max_{y \in Y} (f(y) + \sum_{i,j} u(i, j)y(i, j)) \]

French-to-English

\[ z^* = \arg \max_{z \in Z} (g(z) - \sum_{i,j} u(i, j)z(i, j)) \]

**Key**

- \( f(y) \) \( \iff \) HMM Alignment
- \( g(z) \) \( \iff \) HMM Alignment
- \( Y \) \( \iff \) English-to-French model
- \( Z \) \( \iff \) French-to-English model
- \( y(i, j) = 1 \) if French word \( i \) aligns to English word \( j \)

**Penalties**

\( u(i, j) = 0 \) for all \( i, j \)
\[
\begin{align*}
\text{English-to-French} & \quad f(y) \iff \text{HMM Alignment} & \quad g(z) \iff \text{HMM Alignment} \\
\mathcal{Y} & \iff \text{English-to-French model} & \mathcal{Z} & \iff \text{French-to-English model} \\
y(i,j) = 1 & \text{ if } \text{French word } i \text{ aligns to English word } j
\end{align*}
\]
English-to-French

\[ y^* = \arg \max_{y \in Y} (f(y) + \sum_{i,j} u(i, j)y(i, j)) \]

French-to-English

\[ z^* = \arg \max_{z \in Z} (g(z) - \sum_{i,j} u(i, j)z(i, j)) \]

Key

\[ f(y) \iff \text{HMM Alignment} \]
\[ \mathcal{Y} \iff \text{English-to-French model} \]
\[ y(i, j) = 1 \text{ if French word } i \text{ aligns to English word } j \]
\[ g(z) \iff \text{HMM Alignment} \]
\[ \mathcal{Z} \iff \text{French-to-English model} \]
English-to-French

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j)) \]

French-to-English

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j)) \]

Key

\[ f(y) \iff \text{HMM Alignment} \]
\[ \mathcal{Y} \iff \text{English-to-French model} \]
\[ y(i,j) = 1 \text{ if } \text{French word } i \text{ aligns to English word } j \]

Penalties

\begin{array}{c|c}
\hline
\text{Iteration 1} & \text{u}(i,j) \text{ = 0 for all } i,j \\
\hline
\text{u}(3,2) & -1 \\
\text{u}(2,2) & 1 \\
\text{u}(2,3) & -1 \\
\text{u}(3,3) & 1 \\
\hline
\end{array}
**English-to-French**

$$y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i, j)y(i, j))$$

**French-to-English**

$$z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i, j)z(i, j))$$

**Penalties**

<table>
<thead>
<tr>
<th>$u(i, j)$</th>
<th>Iteration 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u(3, 2)$</td>
<td>-1</td>
</tr>
<tr>
<td>$u(2, 2)$</td>
<td>1</td>
</tr>
<tr>
<td>$u(2, 3)$</td>
<td>-1</td>
</tr>
<tr>
<td>$u(3, 3)$</td>
<td>1</td>
</tr>
</tbody>
</table>

**Key**

- $f(y) \Leftarrow$ HMM Alignment
- $g(z) \Leftarrow$ HMM Alignment
- $\mathcal{Y} \Leftarrow$ English-to-French model
- $\mathcal{Z} \Leftarrow$ French-to-English model
- $y(i, j) = 1$ if French word $i$ aligns to English word $j$
English-to-French

\[ y^* = \arg \max_{y \in Y} \left( f(y) + \sum_{i,j} u(i, j) y(i, j) \right) \]

French-to-English

\[ z^* = \arg \max_{z \in Z} \left( g(z) - \sum_{i,j} u(i, j) z(i, j) \right) \]

Penalties

\[
\begin{array}{c|c}
  i, j & u(i, j) \\
  \hline
  (3, 2) & -1 \\
  (2, 2) & 1 \\
  (2, 3) & -1 \\
  (3, 3) & 1 \\
\end{array}
\]

Iteration 1

\[
\begin{array}{c|c}
  i, j & u(i, j) \\
  \hline
  (3, 2) & -1 \\
  (2, 2) & 1 \\
  (2, 3) & -1 \\
  (3, 3) & 1 \\
\end{array}
\]

Key

\[
\begin{array}{c|c}
  f(y) & \text{HMM Alignment} \\
  \mathcal{Y} & \text{English-to-French model} \\
  g(z) & \text{HMM Alignment} \\
  \mathcal{Z} & \text{French-to-English model} \\
  y(i, j) = 1 & \text{if French word } i \text{ aligns to English word } j \\
\end{array}
\]
English-to-French

\[ y^* = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j)y(i,j)) \]

French-to-English

\[ z^* = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j)z(i,j)) \]

Key

- \( f(y) \) \( \iff \) HMM Alignment
- \( \mathcal{Y} \) \( \iff \) English-to-French model
- \( y(i,j) = 1 \) if French word \( i \) aligns to English word \( j \)
- \( g(z) \) \( \iff \) HMM Alignment
- \( \mathcal{Z} \) \( \iff \) French-to-English model

Penalties

<table>
<thead>
<tr>
<th>( u(i,j) )</th>
<th>Iteration 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u(3,2) )</td>
<td>-1</td>
</tr>
<tr>
<td>( u(2,2) )</td>
<td>1</td>
</tr>
<tr>
<td>( u(2,3) )</td>
<td>-1</td>
</tr>
<tr>
<td>( u(3,3) )</td>
<td>1</td>
</tr>
</tbody>
</table>

\( u(i,j) = 0 \) for all \( i,j \)
English-to-French

\[ y^\ast = \arg \max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i, j)y(i, j)) \]

French-to-English

\[ z^\ast = \arg \max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i, j)z(i, j)) \]

Key

- \( f(y) \iff \text{HMM Alignment} \)
- \( \mathcal{Y} \iff \text{English-to-French model} \)
- \( y(i, j) = 1 \) if French word \( i \) aligns to English word \( j \)

Penalties

\[
\begin{array}{c|c}
\text{Iteration 1} & \\
\hline
u(3, 2) & -1 \\
u(2, 2) & 1 \\
u(2, 3) & -1 \\
u(3, 3) & 1 \\
\end{array}
\]

\( u(i, j) = 0 \) for all \( i, j \)
4. Linear programming

**aim:** explore the connections between Lagrangian relaxation and linear programming

- basic optimization over the simplex
- formal properties of linear programming
- full example with fractional optimal solutions
Simplex

**define:**

- $\Delta_y \subset \mathcal{R}^{\lvert Y \rvert}$ is the simplex over $Y$ where $\alpha \in \Delta_y$ implies
  \[ \alpha_y \geq 0 \text{ and } \sum_y \alpha_y = 1 \]

- $\alpha$ is distribution over $Y$

- $\Delta_z$ is the simplex over $Z$

- $\delta_y : Y \to \Delta_y$ maps elements to the simplex

**example:**

\[ Y = \{ y_1, y_2, y_3 \} \]

vertices

- $\delta_y(y_1) = (1, 0, 0)$
- $\delta_y(y_2) = (0, 1, 0)$
- $\delta_y(y_3) = (0, 0, 1)$
Theorem 1. Simplex linear program

optimize over the simplex $\Delta_y$ instead of the discrete sets $\mathcal{Y}$

goal: optimize linear program

$$\max_{\alpha \in \Delta_y} \sum_{y} \alpha_y f(y)$$

theorem:

$$\max_{y \in \mathcal{Y}} f(y) = \max_{\alpha \in \Delta_y} \sum_{y} \alpha_y f(y)$$

proof: points in $\mathcal{Y}$ correspond to the extreme points of simplex

$$\{\delta_y(y) : y \in \mathcal{Y}\}$$

linear program has optimum at extreme point

proof shows that best distribution chooses a single parse
Combined linear program

optimize over the simplices $\Delta_y$ and $\Delta_z$ instead of the discrete sets $Y$ and $Z$

**goal:** optimize linear program

$$\max_{\alpha \in \Delta_y, \beta \in \Delta_z} \sum_y \alpha_y f(y) + \sum_z \beta_z g(z)$$

such that for all $i, t$

$$\sum_y \alpha_y y(i, t) = \sum_z \beta_z z(i, t)$$

**note:** the two distributions must match in expectation of POS tags

the best distributions $\alpha^*, \beta^*$ are possibly no longer a single parse tree or tag sequence
Lagrangian:

\[ M(u, \alpha, \beta) = \sum_y \alpha_y f(y) + \sum_z \beta_z g(z) + \sum_{i,t} u(i, t) \left( \sum_y \alpha_y y(i, t) - \sum_z \beta_z z(i, t) \right) \]

\[ = \left( \sum_y \alpha_y f(y) + \sum_{i,t} u(i, t) \sum_y \alpha_y y(i, t) \right) + \left( \sum_z \beta_z g(z) - \sum_{i,t} u(i, t) \sum_z \beta_z z(i, t) \right) \]

Lagrangian dual:

\[ M(u) = \max_{\alpha \in \Delta_y, \beta \in \Delta_z} M(u, \alpha, \beta) \]
Theorem 2. Strong duality

**define:**
- $\alpha^*, \beta^*$ is the optimal assignment to $\alpha, \beta$ in the linear program

**theorem:**
\[
\min_u M(u) = \sum_y \alpha^*_y f(y) + \sum_z \beta^*_z g(z)
\]

**proof:** by linear programming duality
Theorem 3. Dual relationship

**Theorem:** for any value of $u$,

$$M(u) = L(u)$$

**Note:** solving the original Lagrangian dual also solves dual of the linear program
Theorem 3. Dual relationship (proof sketch)

focus on $\mathcal{Y}$ term in Lagrangian

\[ L(u) = \max_{y \in \mathcal{Y}} \left( f(y) + \sum_{i,t} u(i, t)y(i, t) \right) + \ldots \]

\[ M(u) = \max_{\alpha \in \Delta_y} \left( \sum_y \alpha_y f(y) + \sum_{i,t} u(i, t) \sum_y \alpha_y y(i, t) \right) + \ldots \]

by theorem 1. optimization over $\mathcal{Y}$ and $\Delta_y$ have the same max

similar argument for $\mathcal{Z}$ gives $L(u) = M(u)$
Summary

\[ f(y) + g(z) \quad \text{original primal objective} \]
\[ L(u) \quad \text{original dual} \]
\[ \sum_y \alpha_y f(y) + \sum_z \beta_z g(z) \quad \text{LP primal objective} \]
\[ M(u) \quad \text{LP dual} \]

relationship between LP dual, original dual, and LP primal objective

\[ \min_u M(u) = \min_u L(u) = \sum_y \alpha^*_y f(y) + \sum_z \beta^*_z g(z) \]
Concrete example

- $\mathcal{Y} = \{y_1, y_2, y_3\}$
- $\mathcal{Z} = \{z_1, z_2, z_3\}$
- $\Delta_y \subset \mathbb{R}^3$, $\Delta_z \subset \mathbb{R}^3$
choose:
- \( \alpha^{(1)} = (0, 0, 1) \in \Delta_y \) is representation of \( y_3 \)
- \( \beta^{(1)} = (0, 0, 1) \in \Delta_z \) is representation of \( z_3 \)

confirm:
\[
\sum_y \alpha_y^{(1)} y(i, t) = \sum_z \beta_z^{(1)} z(i, t)
\]

\( \alpha^{(1)} \) and \( \beta^{(1)} \) satisfy agreement constraint
Fractional solution

\[ y_1 \quad y_2 \quad y_3 \]

\[
\begin{array}{c}
 \text{\(x\)} \\
 a \\
 \text{He is} \\
 \text{\(y\)} \\
 a \\
 \text{He is} \\
 b \\
 \text{He is} \\
 c \\
 \text{He is} \\
 \text{\(z\)} \\
 a &\rightarrow& b \\
 \downarrow & & \downarrow \\
 \text{He is} \\
 b &\rightarrow& a \\
 \downarrow & & \downarrow \\
 \text{He is} \\
 c &\rightarrow& c \\
 \downarrow & & \downarrow \\
 \text{He is} \\
\end{array}
\]

choose:

- \(\alpha^{(2)} = (0.5, 0.5, 0) \in \Delta_y\) is combination of \(y_1\) and \(y_2\)
- \(\beta^{(2)} = (0.5, 0.5, 0) \in \Delta_z\) is combination of \(z_1\) and \(z_2\)

confirm:

\[
\sum_{y} \alpha^{(2)}_y y(i, t) = \sum_{z} \beta^{(2)}_z z(i, t)
\]

\(\alpha^{(2)}\) and \(\beta^{(2)}\) satisfy agreement constraint, but not integral
Optimal solution

weights:
- the choice of $f$ and $g$ determines the optimal solution
- if $(f, g)$ favors $(\alpha^{(2)}, \beta^{(2)})$, the optimal solution is fractional

example: $f = [1 \ 1 \ 2] \text{ and } g = [1 \ 1 \ -2]$
- $f \cdot \alpha^{(1)} + g \cdot \beta^{(1)} = 0 \text{ vs } f \cdot \alpha^{(2)} + g \cdot \beta^{(2)} = 2$
- $\alpha^{(2)}, \beta^{(2)}$ is optimal, even though it is fractional

summary: dual and LP primal optimal:

$$
\min_u M(u) = \min_u L(u) = \sum_y \alpha^{(2)}_y f(y) + \sum_z \beta^{(2)}_z g(z) = 2
$$

original primal optimal:

$$
f(y^*) + g(z^*) = 0
$$
round 1

**dual solutions:**

\[ y_3 \]
\[ x \]
\[ c \]
\[ c \]
\[ He \]
\[ is \]

**dual values:**

\[ y^{(1)} \quad 2.00 \]
\[ z^{(1)} \quad 1.00 \]
\[ L(u^{(1)}) \quad 3.00 \]

**previous solutions:**

\[ y_3 \]
\[ z_2 \]

Round

dual solutions:

\[ x \]
\[ b \rightarrow a \]
\[ c \]
\[ c \]
\[ He \]
\[ is \]

Round

\[ y_3 \]
\[ z_2 \]
round 1

dual solutions:

\[ x \]
\[ c \]
\[ He \]
\[ c \]
\[ y \]
\[ 3 \]
\[ b \]
\[ He \]
\[ a \]
\[ is \]
\[ z \]
\[ 2 \]

round 2

dual solutions:

\[ y_2 \]
\[ z_1 \]
\[ x \]
\[ a \rightarrow b \]
\[ b \]
\[ b \]
\[ He \]
\[ is \]

round 3

dual solutions:

\[ y_1 \]
\[ z_1 \]
\[ a \]
\[ b \]
\[ He \]
\[ is \]

round 4

dual solutions:

\[ y_1 \]
\[ z_1 \]
\[ a \]
\[ b \]
\[ He \]
\[ is \]

round 5

dual solutions:

\[ y_2 \]
\[ z_2 \]
\[ b \]
\[ He \]
\[ a \]
\[ is \]

round 6

dual solutions:

\[ y_1 \]
\[ z_1 \]
\[ a \]
\[ b \]
\[ He \]
\[ is \]

round 7

dual solutions:

\[ y_2 \]
\[ z_2 \]
\[ b \]
\[ He \]
\[ a \]
\[ is \]

round 8

dual solutions:

\[ y_1 \]
\[ z_1 \]
\[ a \]
\[ b \]
\[ He \]
\[ is \]

round 9

dual solutions:

\[ y_2 \]
\[ z_2 \]
\[ b \]
\[ He \]
\[ a \]
\[ is \]

dual values:

\[ y^{(2)} \] 2.00
\[ z^{(2)} \] 1.00
\[ L(u^{(2)}) \] 3.00

previous solutions:

\[ y_3 \] \[ z_2 \]
\[ y_2 \] \[ z_1 \]
round 3

dual solutions:

\begin{align*}
  y_1 & \quad \quad \quad \quad z_1 \\
  x & \quad \quad \quad \quad a \rightarrow b \\
  a & \quad a \\
  \text{He} & \quad \text{is} \\
\end{align*}

dual values:

\begin{align*}
  y^{(3)} & \quad 2.50 \\
  z^{(3)} & \quad 0.50 \\
  L(u^{(3)}) & \quad 3.00 \\
\end{align*}

previous solutions:

\begin{align*}
  y_3 & \quad z_2 \\
  y_2 & \quad z_1 \\
  y_1 & \quad z_1 \\
\end{align*}
round 4

dual solutions:

\[ y_1 \quad z_1 \]

\[ x \quad a \rightarrow b \]

\[ a \quad a \quad \text{He} \quad \text{is} \]

\[ a \quad \text{He} \quad \text{is} \]

\[ y_1 \quad z_1 \quad \text{He} \quad \text{is} \]

dual values:

\[ y^{(4)} = 2.17 \]
\[ z^{(4)} = 0.17 \]
\[ L(u^{(4)}) = 2.33 \]

previous solutions:

\[ y_3 \quad z_2 \]
\[ y_2 \quad z_1 \]
\[ y_1 \quad z_1 \]
\[ y_1 \quad z_1 \]
\[ y_2 \quad z_2 \]
\[ y_1 \quad z_1 \]
\[ y_2 \quad z_2 \]
\[ y_1 \quad z_1 \]
round 5

dual solutions:

\[
\begin{align*}
  x & \quad \text{He} \quad c \\
  y_2 & \quad z_2 \\
  b & \quad b \quad \downarrow \quad \downarrow \\
  \text{He} & \quad \text{is} \quad \text{is}
\end{align*}
\]

dual values:

\[
\begin{align*}
  y^{(5)} & = 2.08 \\
  z^{(5)} & = 0.08 \\
  L(u^{(5)}) & = 2.17
\end{align*}
\]

previous solutions:

\[
\begin{align*}
  y_3 & \quad z_2 \\
  y_2 & \quad z_1 \\
  y_1 & \quad z_1 \\
  y_1 & \quad z_1 \\
  y_2 & \quad z_2
\end{align*}
\]
round 6

dual solutions:

\[ y_1 \quad z_1 \]

\[ x \quad a \quad a \quad \text{He} \quad \text{is} \quad a \rightarrow b \quad \text{He} \quad \text{is} \]

dual values:
\[
\begin{align*}
y^{(6)} & \quad 2.12 \\
z^{(6)} & \quad 0.12 \\
L(u^{(6)}) & \quad 2.23 
\end{align*}
\]

previous solutions:
\[
\begin{align*}
y_3 & \quad z_2 \\
y_2 & \quad z_1 \\
y_1 & \quad z_1 \\
y_1 & \quad z_1 \\
y_2 & \quad z_2 \\
y_1 & \quad z_1 
\end{align*}
\]
round 7

dual solutions:

\[
\begin{align*}
&y_2 \\
&x \\
&b & b \\
&\text{He} \quad \text{is} \\
&b & a \\
&\text{He} \quad \text{is}
\end{align*}
\]

dual values:
\[
\begin{align*}
y^{(7)} & = 2.05 \\
z^{(7)} & = 0.05 \\
L(u^{(7)}) & = 2.10
\end{align*}
\]

previous solutions:
\[
\begin{align*}
y_3 & \quad z_2 \\
y_2 & \quad z_1 \\
y_1 & \quad z_1 \\
y_1 & \quad z_1 \\
y_2 & \quad z_2 \\
y_1 & \quad z_1 \\
y_2 & \quad z_2
\end{align*}
\]
round 8

dual solutions:

\[ \begin{align*}
  x & \quad y_1 \\
  a & \quad a \\
  \text{He} & \quad \text{is}
\end{align*} \]

\[ \begin{align*}
  z_1 & \\
  a & \rightarrow b \\
  \text{He} & \quad \text{is}
\end{align*} \]

dual values:

\[ \begin{align*}
  y^{(8)} & \quad 2.09 \\
  z^{(8)} & \quad 0.09 \\
  L(u^{(8)}) & \quad 2.19
\end{align*} \]

previous solutions:

\[ \begin{align*}
  y_3 & \quad z_2 \\
  y_2 & \quad z_1 \\
  y_1 & \quad z_1 \\
  y_1 & \quad z_1 \\
  y_2 & \quad z_2 \\
  y_1 & \quad z_1 \\
  y_1 & \quad z_1
\end{align*} \]
round 9

dual solutions:

\[ y_2 \quad z_2 \]

\[ x \quad b \rightarrow a \]

\[ b \quad b \quad \text{He is} \quad \text{He is} \]

dual values:

\[ y^{(9)} \quad 2.03 \]
\[ z^{(9)} \quad 0.03 \]
\[ L(u^{(9)}) \quad 2.06 \]

previous solutions:

\[ y_3 \quad z_2 \]
\[ y_2 \quad z_1 \]
\[ y_1 \quad z_1 \]
\[ y_1 \quad z_1 \]
\[ y_2 \quad z_2 \]
\[ y_1 \quad z_1 \]
\[ y_2 \quad z_2 \]
\[ y_1 \quad z_1 \]
\[ y_2 \quad z_2 \]
5. Practical issues

tracking the progress of the algorithm

- know current dual value and (possibly) primal value

choice of update rate $\alpha_k$

- various strategies; success with rate based on dual progress

lazy update of dual solutions

- if updates are sparse, can avoid dynamically update solutions

extracting solutions if algorithm does not converge

- best primal feasible solution; average solutions
 Phrase-Based Translation

define:

- source-language sentence words \( x_1, \ldots, x_N \)
- phrase translation \( p = (s, e, t) \)
- translation derivation \( y = p_1, \ldots, p_L \)

example:

\[
\begin{array}{ccccccc}
\phantom{\text{das muss uns...}} \ \\
\text{x1} & \text{x2} & \text{x3} & \text{x4} & \text{x5} & \text{x6} \\
\end{array}
\]

\[
\text{das muss unsere sorge gleichermaßen sein}
\]
Phrase-Based Translation

define:
- source-language sentence words $x_1, \ldots, x_N$
- phrase translation $p = (s, e, t)$
- translation derivation $y = p_1, \ldots, p_L$

example:

$y = \{(1, 2, \text{this must}),\}$
Phrase-Based Translation

define:
▶ source-language sentence words \( x_1, \ldots, x_N \)
▶ phrase translation \( p = (s, e, t) \)
▶ translation derivation \( y = p_1, \ldots, p_L \)

element:

\[
y = \{(1, 2, \text{this must}), (5, 5, \text{also}), \}
\]
Phrase-Based Translation

define:
- source-language sentence words $x_1, \ldots, x_N$
- phrase translation $p = (s, e, t)$
- translation derivation $y = p_1, \ldots, p_L$

example:

$y = \{(1, 2, \text{this must}), (5, 5, \text{also}), (6, 6, \text{be})\}$
Phrase-Based Translation

define:

- source-language sentence words $x_1, \ldots, x_N$
- phrase translation $p = (s, e, t)$
- translation derivation $y = p_1, \ldots, p_L$

example:

$y = \{(1, 2, \text{this must}), (5, 5, \text{also}), (6, 6, \text{be}), (3, 4, \text{our concern})\}$
Phrase-Based Translation

define:
- source-language sentence words $x_1, \ldots, x_N$
- phrase translation $p = (s, e, t)$
- translation derivation $y = p_1, \ldots, p_L$

example:

$$y = \{(1, 2, \text{this must}), (5, 5, \text{also}), (6, 6, \text{be}), (3, 4, \text{our concern})\}$$
Phrase-Based Translation

define:

- source-language sentence words $x_1, \ldots, x_N$
- phrase translation $p = (s, e, t)$
- translation derivation $y = p_1, \ldots, p_L$

example:

$y = \{(1, 2, \text{this must}), (5, 5, \text{also}), (6, 6, \text{be}), (3, 4, \text{our concern})\}$
Scoring Derivations

derivation:

\[ y = \{(1, 2, \text{this must}), (5, 5, \text{also}), (6, 6, \text{be}), (3, 4, \text{our concern})\} \]

objective:

\[
f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta|t(p_k) + 1 - s(p_{k+1})|\]

- language model score \( h \)
- phrase translation score \( g \)
- distortion penalty \( \eta \)
Scoring Derivations

derivation:

\[ y = \{ (1, 2, \text{this must}), (5, 5, \text{also}), (6, 6, \text{be}), (3, 4, \text{our concern}) \} \]

objective:

\[
f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta |t(p_k) + 1 - s(p_{k+1})|
\]

- language model score \( h \)
- phrase translation score \( g \)
- distortion penalty \( \eta \)
Scoring Derivations

derivation:

\[ y = \{(1, 2, \text{this must}), (5, 5, \text{also}), (6, 6, \text{be}), (3, 4, \text{our concern})\} \]

objective:

\[
f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta |t(p_k) + 1 - s(p_{k+1})|
\]

- language model score \( h \)
- phrase translation score \( g \)
- distortion penalty \( \eta \)
Scoring Derivations

derivation:

\[ y = \{(1, 2, \text{this must}), (5, 5, \text{also}), (6, 6, \text{be}), (3, 4, \text{our concern})\} \]

objective:

\[
f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta |t(p_k) + 1 - s(p_{k+1})|
\]

- language model score \( h \)
- phrase translation score \( g \)
- distortion penalty \( \eta \)
Relaxed Problem

\( \mathcal{Y}' \): only requires the total number of words translated to be \( N \)

\[
\mathcal{Y}' = \{ y : \sum_{i=1}^{N} y(i) = N \text{ and the distortion limit } d \text{ is satisfied} \}
\]

element:

\[
y(i) \quad 0 \quad 1 \quad 2 \quad 2 \quad 0 \quad 1
\]

\[
\begin{array}{cccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
das & muss & unsere sorge & gleichermass \text{en} & sein
\end{array}
\]

(3, 4, our concern)(2, 2, must)(6, 6, be)(3, 4, our concern)
Relaxed Problem

\( \mathcal{Y}' \): only requires the total number of words translated to be \( N \)

\[ \mathcal{Y}' = \{ y : \sum_{i=1}^{N} y(i) = N \text{ and the distortion limit } d \text{ is satisfied} \} \]

example:

\[
\begin{align*}
  y(i) &\quad 0 &\quad 1 &\quad 2 &\quad 2 &\quad 0 &\quad 1 \\
  x_1 &\quad \text{das} &\quad \text{muss} &\quad \text{unsere} &\quad \text{sorge} &\quad \text{gleichermaßen} &\quad \text{sein} \\
  x_2 &\quad &\quad &\quad &\quad &\quad &\quad \\
  x_3 &\quad &\quad &\quad &\quad &\quad &\quad \\
  x_4 &\quad &\quad &\quad &\quad &\quad &\quad \\
  x_5 &\quad &\quad &\quad &\quad &\quad &\quad \\
  x_6 &\quad &\quad &\quad &\quad &\quad &\quad
\end{align*}
\]

(3, 4, our concern)(2, 2, must)(6, 6, be)(3, 4, our concern)
Relaxed Problem

\( \mathcal{Y}'\): only requires the total number of words translated to be \( N \)

\[
\mathcal{Y}' = \{ y : \sum_{i=1}^{N} y(i) = N \text{ and the distortion limit } d \text{ is satisfied} \}
\]

example:

\[
y(i) = \begin{bmatrix} 0 & 1 & 2 & 2 & 0 & 1 \\
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}
\]

\[
\begin{align*}
\text{das} &\quad \text{muss} &\quad \text{unsere sorge} &\quad \text{gleichermaßen} &\quad \text{sein} \\
\text{our concern} &\quad \text{must} &\quad \text{be} &\quad \text{our concern} \\
(3, 4, \text{our concern}) &\quad (2, 2, \text{must}) &\quad (6, 6, \text{be}) &\quad (3, 4, \text{our concern})
\end{align*}
\]
Relaxed Problem

\( \mathcal{Y}'\): only requires the total number of words translated to be \( N \)

\[
\mathcal{Y}' = \{ y : \sum_{i=1}^{N} y(i) = N \text{ and the distortion limit } d \text{ is satisfied} \}
\]

example:

\[
y(i) \quad 0 \quad 1 \quad 2 \quad 2 \quad 0 \quad 1 \quad \text{sum} \quad 6
\]

\[
\begin{array}{cccccc}
& \text{das} & \text{muss} & \text{unsere sorge} & \text{gleichermaßen} & \text{sein} \\
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\text{our concern} & \text{must} & \text{be} & \text{our concern}
\end{array}
\]

\((3, 4, \text{our concern})(2, 2, \text{must})(6, 6, \text{be})(3, 4, \text{our concern})\)
original:

$$\arg \max_{y \in \mathcal{Y}} f(y)$$

$$\mathcal{Y} = \{y : y(i) = 1 \ \forall i = 1 \ldots N\}$$

rewrite:

$$\arg \max_{y \in \mathcal{Y}'} f(y) \text{ such that } y(i) = 1 \ \forall i = 1 \ldots N$$

$$\mathcal{Y}' = \{y : \sum_{i=1}^{N} y(i) = N\}$$

exact DP is NP-hard

$$\mathcal{Y} = \{y : y(i) = 1 \ \forall i = 1 \ldots N\} \ldots \text{ that } y(i) = 1 \ \forall i = 1 \ldots N$$

using Lagrangian relaxation

$$\mathcal{Y}' = \{y : \sum_{i=1}^{N} y(i) = N\} \sum \text{ to } N$$
Lagrangian Relaxation Method

original:

\[
\arg\max_{y \in \mathcal{Y}} f(y)
\]

exact DP is NP-hard

\[\mathcal{Y} = \{y : y(i) = 1 \ \forall \ i = 1 \ldots N\}\]

rewrite:

\[
\arg\max_{y \in \mathcal{Y}'} f(y) \quad \text{such that} \quad y(i) = 1 \ \forall \ i = 1 \ldots N
\]

\[\mathcal{Y}' = \{y : \sum_{i=1}^{N} y(i) = N\}\]

sum to $N$
Lagrangian Relaxation Method

original:
\[
\arg \max_{y \in \mathcal{Y}} f(y)
\]
exact DP is NP-hard

\[\mathcal{Y} = \{y : y(i) = 1 \ \forall i = 1 \ldots N\}\]

rewrite:
\[
\arg \max_{y \in \mathcal{Y}'} f(y) \text{ such that } y(i) = 1 \ \forall i = 1 \ldots N
\]

\[\mathcal{Y}' = \{y : \sum_{i=1}^{N} y(i) = N\}\]
Lagrangian Relaxation Method

original:

\[
\arg\max_{y \in \mathcal{Y}} f(y)
\]

exact DP is NP-hard

\[
\mathcal{Y} = \{y : y(i) = 1 \forall i = 1 \ldots N\}
\]

rewrite:

\[
\arg\max_{y \in \mathcal{Y}'} f(y) \quad \text{such that} \quad y(i) = 1 \forall i = 1 \ldots N
\]

can be solved efficiently by DP

\[
\mathcal{Y}' = \{y : \sum_{i=1}^{N} y(i) = N\}
\]

sum to \( N \)
original:

\[
\arg \max_{y \in \mathcal{Y}} f(y)
\]

exact DP is NP-hard

\[
\mathcal{Y} = \{ y : y(i) = 1 \ \forall \ i = 1 \ldots N \}
\]

rewrite:

\[
\arg \max_{y \in \mathcal{Y}'} f(y) \quad \text{such that} \quad y(i) = 1 \ \forall \ i = 1 \ldots N
\]

can be solved efficiently by DP

\[
\mathcal{Y}' = \{ y : \sum_{i=1}^{N} y(i) = N \}
\]

using Lagrangian relaxation

\[
\text{sum to } N
\]
Algorithm

Iteration 1:

- update $u(i)$: $u(i) \leftarrow u(i) - \alpha(y(i) - 1)$
  
  $\alpha = 1$

<table>
<thead>
<tr>
<th>$u(i)$</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>$y(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
</tr>
<tr>
<td>$x_2$</td>
</tr>
<tr>
<td>$x_3$</td>
</tr>
<tr>
<td>$x_4$</td>
</tr>
<tr>
<td>$x_5$</td>
</tr>
<tr>
<td>$x_6$</td>
</tr>
</tbody>
</table>

das muss unsere sorge gleichermaßen sein
Algorithm

Iteration 1:

- update $u(i)$: $u(i) \leftarrow u(i) - \alpha(y(i) - 1)$

$\alpha = 1$

<table>
<thead>
<tr>
<th>$u(i)$</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y(i)$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\begin{align*}
    x_1 & \quad \text{das} \\
    x_2 & \quad \text{muss} \\
    x_3 & \quad \text{unsere} \\
    x_4 & \quad \text{sorge} \\
    x_5 & \quad \text{gleichermaßen} \\
    x_6 & \quad \text{sein} \\
\end{align*}$

our concern must be our concern
Algorithm

Iteration 1:

- update $u(i)$: $u(i) \leftarrow u(i) - \alpha(y(i) - 1)$

$$\alpha = 1$$

<table>
<thead>
<tr>
<th>$u(i)$</th>
<th>1</th>
<th>0</th>
<th>-1</th>
<th>-1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y(i)$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$u(i)$ update

das muss unsere sorge gleichermaßen sein
our concern our concern must be
Algorithm

Iteration 2:

- update $u(i)$: $u(i) \leftarrow u(i) - \alpha(y(i) - 1)$

$\alpha = 0.5$

$u(i)$

\[
\begin{array}{cccccc}
1 & 0 & -1 & -1 & 1 & 0 \\
\end{array}
\]

$y(i)$

\[
\begin{array}{cccccc}
\hat{x}_1 & \hat{x}_2 & \hat{x}_3 & \hat{x}_4 & \hat{x}_5 & \hat{x}_6 \\
\end{array}
\]

das muss unsere sorge gleichermaßen sein
Algorithm

Iteration 2:

- update $u(i)$: $u(i) \leftarrow u(i) - \alpha(y(i) - 1)$

$\alpha = 0.5$

$u(i)$ | 1 | 0 | −1 | −1 | 1 | 0
---|---|---|---|---|---|---
$y(i)$ | 1 | 2 | 0 | 0 | 2 | 1

This must be equally muss unsere sorge gleichermäßen sein
Algorithm

Iteration 2:

▶ update $u(i)$: $u(i) \leftarrow u(i) - \alpha(y(i) - 1)$

$\alpha = 0.5$

| $u(i)$ | 1 | 0.5 | 0.5 | 0.5 | 0.5 | 0 |
| $y(i)$ | 1 | 2 | 0 | 0 | 2 | 1 |

$\chi_1$ | $\chi_2$ | $\chi_3$ | $\chi_4$ | $\chi_5$ | $\chi_6$

das | muss | unsere | sorge | gleichermäßen | sein

das muss unsere sorge gleichermäßen sein

this must be equally must equally
Algorithm

Iteration 3:

- update $u(i)$: $u(i) \leftarrow u(i) - \alpha(y(i) - 1)$

$\alpha = 0.5$

$\begin{array}{cccccc}
u(i) & 1 & -0.5 & -0.5 & -0.5 & 0.5 & 0 \\
\end{array}$

$\begin{array}{cccccc}
y(i) & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\end{array}$

das muss unsere sorge gleichermaßen sein
Algorithm

Iteration 3:

- update $u(i)$: $u(i) \leftarrow u(i) - \alpha(y(i) - 1)$

\[ \alpha = 0.5 \]

<table>
<thead>
<tr>
<th>$u(i)$</th>
<th>1</th>
<th>-0.5</th>
<th>-0.5</th>
<th>-0.5</th>
<th>0.5</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y(i)$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$x_1$  x_2  x_3  x_4  x_5  x_6$

das  muss  unsere  sorge  gleichermassen  sein

das muss unsere sorge gleichermassen sein

this must also be our concern
Tightening the Relaxation

In some cases, we never reach \( y(i) = 1 \) for \( i = 1 \ldots N \)

If dual \( L(u) \) is not decreasing fast enough

run for 10 more iterations

count number of times each constraint is violated

add 3 most often violated constraints
Tightening the Relaxation

Iteration 41:

\[
\begin{align*}
\text{count}(i) & : 0 & 0 & 0 & 0 & 1 & 1 \\
y(i) & : 1 & 1 & 1 & 1 & 2 & 0 \\
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\text{das muss} & \quad \text{unsere sorge} & \quad \text{gleichermaßen} \\
\text{this must} & \quad \text{also} & \quad \text{our concern} & \quad \text{equally} \\
\end{align*}
\]

this must also our concern equally

Add 2 hard constraints \((x_5, x_6)\) to the dynamic program.
Tightening the Relaxation

Iteration 42:

\[
\begin{align*}
\text{count}(i) & : 0 & 0 & 0 & 0 & 2 & 2 \\
y(i) & : 1 & 1 & 1 & 1 & 0 & 2 \\
\end{align*}
\]

\[
\begin{align*}
&x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
das muss & & unsere sorge & & gleichermaßen & & sein \\
this must & & be & our concern & & is \\
\end{align*}
\]
Tightening the Relaxation

Iteration 43:

\[
\begin{array}{ccccccc}
\text{count}(i) & 0 & 0 & 0 & 0 & 3 & 3 \\
\text{y}(i) & 1 & 1 & 1 & 1 & 2 & 0 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
das & muss & unsere & sorge & gleichermassen & sein \\
this & must & also & our & concern & equally \\
\end{array}
\]
Tightening the Relaxation

Iteration 44:

<table>
<thead>
<tr>
<th>$count(i)$</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>4</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y(i)$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

$x_1$ $x_2$ $x_3$ $x_4$ $x_5$ $x_6$

```
das muss unsere sorge gleichermaßen sein
```

this must be our concern is

Add 2 hard constraints $(x_5, x_6)$ to the dynamic program.
Tightening the Relaxation

Iteration 50:

\[
\begin{array}{ccccccc}
\text{count}(i) & 0 & 0 & 0 & 0 & 10 & 10 \\
\text{y}(i) & 1 & 1 & 1 & 1 & 2 & 0 \\
\end{array}
\]

\[x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6\]

das muss unsere sorge gleichermaßen sein
this must also our concern equally

Add 2 hard constraints (x_5, x_6) to the dynamic program.
Add 2 hard constraints \((x_5, x_6)\) to the dynamic program
Tightening the Relaxation

Iteration 51:

\[ y(i) = \begin{array}{ccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ \hline das & muss & unsere & sorge & gleichermaßen & sein \\ this & must & also & be & our & concern \end{array} \]

Add 2 hard constraints \((x_5, x_6)\) to the dynamic program.
Experiments: German to English

- Europarl data: German to English
- Test on 1,824 sentences with length 1-50 words
- Converged: 1,818 sentences (99.67%)
Experiments: Number of Iterations

Percentage

Maximum Number of Lagrangian Relaxation Iterations

1-10 words
11-20 words
21-30 words
31-40 words
41-50 words
all

Maximum Number of Lagrangian Relaxation Iterations

Percentage
Experiments: Mean Time in Seconds

<table>
<thead>
<tr>
<th># words</th>
<th>1-10</th>
<th>11-20</th>
<th>21-30</th>
<th>31-40</th>
<th>41-50</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>0.8</td>
<td>10.9</td>
<td>57.2</td>
<td>203.4</td>
<td>679.9</td>
<td>120.9</td>
</tr>
<tr>
<td>median</td>
<td>0.7</td>
<td>8.9</td>
<td>48.3</td>
<td>169.7</td>
<td>484.0</td>
<td>35.2</td>
</tr>
</tbody>
</table>
## Comparison to ILP Decoding

<table>
<thead>
<tr>
<th></th>
<th>(sec.)</th>
<th>(sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>275.2</td>
<td>132.9</td>
</tr>
<tr>
<td>11-15</td>
<td>2,707.8</td>
<td>1,138.5</td>
</tr>
<tr>
<td>16-20</td>
<td>20,583.1</td>
<td>3,692.6</td>
</tr>
</tbody>
</table>
Summary

presented Lagrangian relaxation as a method for decoding in NLP

formal guarantees

• gives certificate or approximate solution
• can improve approximate solutions by tightening relaxation

efficient algorithms

• uses fast combinatorial algorithms
• can improve speed with lazy decoding

widely applicable

• demonstrated algorithms for a wide range of NLP tasks (parsing, tagging, alignment, mt decoding)
Higher-order non-projective dependency parsing

setup: given a model for higher-order non-projective dependency parsing (sibling features)

problem: find non-projective dependency parse that maximizes the score of this model

difficulty:

- model is NP-hard to decode
- complexity of the model comes from enforcing combinatorial constraints

strategy: design a decomposition that separates combinatorial constraints from direct implementation of the scoring function
Non-Projective Dependency Parsing

Important problem in many languages.

Problem is **NP-Hard** for all but the simplest models.
Dual Decomposition

A classical technique for constructing decoding algorithms.

Solve complicated models

$$y^* = \arg \max_y f(y)$$

by decomposing into smaller problems.

Upshot: Can utilize a toolbox of combinatorial algorithms.

- Dynamic programming
- Minimum spanning tree
- Shortest path
- Min-Cut
- ...
Non-Projective Dependency Parsing

*0  John₁  saw₂  a₃  movie₄  today₅  that₆  he₇  liked₈

▶ Starts at the root symbol *
▶ Each word has a exactly one parent word
▶ Produces a tree structure (no cycles)
▶ Dependencies can cross
$f(y) =$

Arc-Factored
Arc-Factored

\[ f(y) = score(\text{head} = \ast_0, \text{mod} = \text{saw}_2) \]
Arc-Factored

\[ f(y) = \text{score}(\text{head} = \ast_0, \text{mod} = \text{saw}_2) + \text{score}(\text{saw}_2, \text{John}_1) \]
Arc-Factored

\[
f(y) = \text{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) + \text{score}(\text{saw}_2, \text{John}_1) + \text{score}(\text{saw}_2, \text{movie}_4) + \text{score}(\text{movie}_4, \text{a}_3) + \text{score}(\text{a}_3, \text{today}_5) + \text{score}(\text{today}_5, \text{that}_6) + \text{score}(\text{that}_6, \text{he}_7) + \text{score}(\text{he}_7, \text{liked}_8)
\]

\[
f(y) = \text{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) + \text{score}(\text{saw}_2, \text{John}_1)
\]

\[
\text{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) = \log p(\text{saw}_2 | *_0)
\]

\[
\text{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) = w \cdot \phi(\text{saw}_2, *_0)
\]

\[
y^* = \arg \max_y f(y) \iff \text{Minimum Spanning Tree Algorithm}
\]
Arc-Factored

\[ f(y) = score(\text{head} = *_0, \text{mod} = \text{saw}_2) + score(\text{saw}_2, \text{John}_1) \]
\[ + score(\text{saw}_2, \text{movie}_4) + score(\text{saw}_2, \text{today}_5) \]
Arc-Factored

\[ f(y) = \text{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) + \text{score}(\text{saw}_2, \text{John}_1) \]

\[ + \text{score}(\text{saw}_2, \text{movie}_4) + \text{score}(\text{saw}_2, \text{today}_5) \]

\[ + \text{score}(\text{movie}_4, \text{a}_3) + ... \]
Arc-Factored

\[ f(y) = \text{score}(\text{head} = *, \text{mod} = \text{saw}_2) + \text{score}(\text{saw}_2, \text{John}_1) \]
\[ + \text{score}(\text{saw}_2, \text{movie}_4) + \text{score}(\text{saw}_2, \text{today}_5) \]
\[ + \text{score}(\text{movie}_4, \text{a}_3) + \ldots \]

\( \text{e.g. } \text{score}(\ast_0, \text{saw}_2) = \log p(\text{saw}_2 | \ast_0) \) (generative model)
**Arc-Factored**

\[
f(y) = \text{score}(\text{head} = \ast_0, \text{mod} = \text{saw}_2) + \text{score}(\text{saw}_2, \text{John}_1) \\
+ \text{score}(\text{saw}_2, \text{movie}_4) + \text{score}(\text{saw}_2, \text{today}_5) \\
+ \text{score}(\text{movie}_4, \text{a}_3) + \ldots
\]

e.g. \( \text{score}(\ast_0, \text{saw}_2) = \log p(\text{saw}_2|\ast_0) \) \hspace{1cm} (generative model)

or \( \text{score}(\ast_0, \text{saw}_2) = w \cdot \phi(\text{saw}_2, \ast_0) \) \hspace{1cm} (CRF/perceptron model)
Arc-Factored

\[ f(y) = \text{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) + \text{score}(\text{saw}_2, \text{John}_1) + \text{score}(\text{saw}_2, \text{movie}_4) + \text{score}(\text{saw}_2, \text{today}_5) + \text{score}(\text{movie}_4, \text{a}_3) + \ldots \]

e.g. \[ \text{score}(*_0, \text{saw}_2) = \log p(\text{saw}_2 | *_0) \] (generative model)

or \[ \text{score}(*_0, \text{saw}_2) = w \cdot \phi(\text{saw}_2, *_0) \] (CRF/perceptron model)

\[ y^* = \arg \max_y f(y) \iff \text{Minimum Spanning Tree Algorithm} \]
Sibling Models

\[ f(y) = \text{score(head = } \ast^0, \text{prev = NULL, mod = } \ldots = \log p(\text{today}|\text{saw}, \text{movie}) \text{ or score(saw, movie, today) = w · } \phi(\text{saw}, \text{movie}, \text{today}) \]

\[ \text{y}^* = \text{arg max}_{\text{y}} f(y) \Rightarrow \text{NP-Hard} \]
Sibling Models

\[ f(y) = \text{score}(\text{head} = *_0, \text{prev} = \text{NULL}, \text{mod} = \text{saw}_2) \]
Sibling Models

\[ f(y) = \text{score}(\text{head} = \ast_0, \text{prev} = \text{NULL}, \text{mod} = \text{saw}_2) \]
\[ + \text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) \]
Sibling Models

\[ f(y) = score(head = *0, prev = NULL, mod = saw2) + score(saw2, NULL, John1) + score(saw2, NULL, movie4) \]
Sibling Models

\[ f(y) = \text{score}(\text{head} = *0, \text{prev} = \text{NULL}, \text{mod} = \text{saw}_2) + \text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) + \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) + \ldots \]
Sibling Models

\[ f(y) = \text{score}(\text{head} = {\ast}_0, \text{prev} = \text{NULL}, \text{mod} = \text{saw}_2) \]

\[ + \text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) \]

\[ + \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) + \ldots \]

e.g. \[ \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) = \log p(\text{today}_5|\text{saw}_2, \text{movie}_4) \]
**Sibling Models**

\[ f(y) = \text{score}(\text{head} = \ast_0, \text{prev} = \text{NULL}, \text{mod} = \text{saw}_2) \]

\[ + \text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) \]

\[ + \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) + \ldots \]

e.g. \[ \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) = \log p(\text{today}_5|\text{saw}_2, \text{movie}_4) \]

or \[ \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) = w \cdot \phi(\text{saw}_2, \text{movie}_4, \text{today}_5) \]
Sibling Models

\[ f(y) = \text{score}(\text{head} = *_0, \text{prev} = \text{NULL}, \text{mod} = \text{saw}_2) \]
\[ + \text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) \]
\[ + \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) + \ldots \]

e.g. \[ \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) = \log p(\text{today}_5 | \text{saw}_2, \text{movie}_4) \]
\[ \text{or score}(\text{saw}_2, \text{movie}_4, \text{today}_5) = w \cdot \phi(\text{saw}_2, \text{movie}_4, \text{today}_5) \]

\[ y^* = \arg \max_y f(y) \Leftarrow \text{NP-Hard} \]
Thought Experiment: Individual Decoding

\[ \begin{align*}
*_{0} & \text{ John}_{1} \text{ saw}_{2} \text{ a}_{3} \text{ movie}_{4} \text{ today}_{5} \text{ that}_{6} \text{ he}_{7} \text{ liked}_{8} \\
\end{align*} \]
Thought Experiment: Individual Decoding

*0 John1 saw2 a3 movie4 today5 that6 he7 liked8

\[
\text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) \\
+ \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5)
\]
Thought Experiment: Individual Decoding

\[ \text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) \]
\[ + \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) \]

\[ = \text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{that}_6) \]
Thought Experiment: Individual Decoding

*0 John1 saw2 a3 movie4 today5 that6 he7 liked8

\[ \text{score}(\text{saw2}, \text{NULL}, \text{John}_1) + \text{score}(\text{saw2}, \text{NULL}, \text{movie}_4) \]
\[ + \text{score}(\text{saw2}, \text{movie}_4, \text{today}_5) \]

\[ = \text{score}(\text{saw2}, \text{NULL}, \text{John}_1) + \text{score}(\text{saw2}, \text{NULL}, \text{that}_6) \]
\[ + \text{score}(\text{saw2}, \text{NULL}, \text{a}_3) + \text{score}(\text{saw2}, \text{a}_3, \text{he}_7) \]
Thought Experiment: Individual Decoding

\[ \text{score}(\text{saw}, \text{NULL}, \text{John}) + \text{score}(\text{saw}, \text{NULL}, \text{movie}) + \text{score}(\text{saw}, \text{movie}, \text{today}) + \text{score}(\text{saw}, \text{NULL}, \text{that}) + \text{score}(\text{saw}, \text{a}, \text{he}) \]

\(2^{n-1}\) possibilities

Under Sibling Model, can solve for each word with Viterbi decoding.
Thought Experiment: Individual Decoding

*0 John1 saw2 a3 movie4 today5 that6 he7 liked8

\[ \text{score}(\text{saw2, NULL, John1}) + \text{score}(\text{saw2, NULL, movie4}) + \text{score}(\text{saw2, movie4, today5}) \]

\[ \text{score}(\text{saw2, NULL, John1}) + \text{score}(\text{saw2, NULL, that6}) \]

\[ \text{score}(\text{saw2, NULL, a3}) + \text{score}(\text{saw2, a3, he7}) \]

\(2^{n-1}\) possibilities

Under Sibling Model, can solve for each word with Viterbi decoding.
Thought Experiment Continued

Idea: Do individual decoding for each head word using dynamic programming.

If we’re lucky, we’ll end up with a valid final tree.
Thought Experiment Continued

Idea: Do individual decoding for each head word using dynamic programming.

If we’re lucky, we’ll end up with a valid final tree.
Thought Experiment Continued

Idea: Do individual decoding for each head word using dynamic programming.

If we’re lucky, we’ll end up with a valid final tree.
Thought Experiment Continued

Idea: Do individual decoding for each head word using dynamic programming.

If we’re lucky, we’ll end up with a valid final tree.
Thought Experiment Continued

Idea: Do individual decoding for each head word using dynamic programming.

If we’re lucky, we’ll end up with a valid final tree.
Thought Experiment Continued

Idea: Do individual decoding for each head word using dynamic programming.

If we’re lucky, we’ll end up with a valid final tree.
Thought Experiment Continued

Idea: Do individual decoding for each head word using dynamic programming.

If we’re lucky, we’ll end up with a valid final tree.
Thought Experiment Continued

Idea: Do individual decoding for each head word using dynamic programming.

If we’re lucky, we’ll end up with a valid final tree.
Thought Experiment Continued

Idea: Do individual decoding for each head word using dynamic programming.

If we’re lucky, we’ll end up with a valid final tree.

But we might violate some constraints.
Goal \( y^* = \arg \max_{y \in Y} f(y) \)
Dual Decomposition Structure

Goal $y^* = \arg \max_{y \in \mathcal{Y}} f(y)$

Rewrite as $\arg\max_{z \in \mathcal{Z}, y \in \mathcal{Y}} f(z) + g(y)$ such that $z = y$
Dual Decomposition Structure

Goal: \( y^* = \arg \max_{y \in \mathcal{Y}} f(y) \)

Rewrite as: \( \arg \max_{z \in \mathcal{Z}, \ y \in \mathcal{Y}} f(z) + g(y) \), such that \( z = y \)

Valid Trees:
- All Possible
- Sibling Arc-Factored

Constraint
Dual Decomposition Structure

Goal \( y^* = \arg \max_{y \in Y} f(y) \)

Rewrite as \( \arg \max_{z \in Z, \ y \in Y} f(z) + g(y) \)

such that \( z = y \)

Valid Trees

All Possible

Valid Trees
Dual Decomposition Structure

Goal: \( y^* = \arg \max_{y \in \mathcal{Y}} f(y) \)

Rewrite as: \( \arg \max_{z \in \mathcal{Z}, y \in \mathcal{Y}} f(z) + g(y) \) such that \( z = y \)

- Sibling
- All Possible
- Valid Trees

Valid Trees: All Possible

Constraint:
Goal $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$

Rewrite as $\arg\max_{z \in \mathcal{Z}, y \in \mathcal{Y}} f(z) + g(y)$ such that $z = y$

Valid Trees

All Possible

Sibling

Arc-Factored

Valid Trees
Dual Decomposition Structure

Goal $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$

Rewrite as $\arg\max_{z \in \mathcal{Z}, y \in \mathcal{Y}} f(z) + g(y)$ such that $z = y$

Valid Trees

All Possible

Constraint

Sibling

Arc-Factored

Constraint
Algorithm Sketch

Set penalty weights equal to 0 for all edges.

For $k = 1$ to $K$

If $y(k)(i,j) = z(k)(i,j)$ for all $i,j$ Return $(y(k),z(k))$
Algorithm Sketch

Set penalty weights equal to 0 for all edges.

For $k = 1$ to $K$

$z^{(k)} ← \text{Decode } (f(z) + \text{penalty}) \text{ by Individual Decoding}$

$y^{(k)} ← \text{Decode } (g(y) - \text{penalty}) \text{ by Minimum Spanning Tree}$

If $y^{(k)}(i,j) = z^{(k)}(i,j)$ for all $i,j$ Return $(y^{(k)}, z^{(k)})$
Algorithm Sketch

Set penalty weights equal to 0 for all edges.

For $k = 1$ to $K$

\[ z^{(k)} \leftarrow \text{Decode } (f(z) + \text{penalty}) \text{ by Individual Decoding} \]

\[ y^{(k)} \leftarrow \text{Decode } (g(y) - \text{penalty}) \text{ by Minimum Spanning Tree} \]
Set penalty weights equal to 0 for all edges.

For $k = 1$ to $K$

$$z^{(k)} \leftarrow \text{Decode } (f(z) + \text{penalty}) \text{ by Individual Decoding}$$

$$y^{(k)} \leftarrow \text{Decode } (g(y) - \text{penalty}) \text{ by Minimum Spanning Tree}$$

If $y^{(k)}(i,j) = z^{(k)}(i,j)$ for all $i,j$ Return $(y^{(k)}, z^{(k)})$
Algorithm Sketch

Set penalty weights equal to 0 for all edges.

For $k = 1$ to $K$

$z^{(k)} \leftarrow$ Decode ($f(z) + \text{penalty}$) by Individual Decoding

$y^{(k)} \leftarrow$ Decode ($g(y) - \text{penalty}$) by Minimum Spanning Tree

If $y^{(k)}(i,j) = z^{(k)}(i,j)$ for all $i,j$ Return $(y^{(k)}, z^{(k)})$

Else Update penalty weights based on $y^{(k)}(i,j) - z^{(k)}(i,j)$
Individual Decoding

\[ *_0 \quad \text{John}_1 \quad \text{saw}_2 \quad \text{a}_3 \quad \text{movie}_4 \quad \text{today}_5 \quad \text{that}_6 \quad \text{he}_7 \quad \text{liked}_8 \]

\[ z^* = \arg \max_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

Minimum Spanning Tree

\[ *_0 \quad \text{John}_1 \quad \text{saw}_2 \quad \text{a}_3 \quad \text{movie}_4 \quad \text{today}_5 \quad \text{that}_6 \quad \text{he}_7 \quad \text{liked}_8 \]

\[ y^* = \arg \max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

Key

\[ f(z) \quad \Leftarrow \quad \text{Sibling Model} \quad \quad g(y) \quad \Leftarrow \quad \text{Arc-Factored Model} \]

\[ \mathcal{Z} \quad \Leftarrow \quad \text{No Constraints} \quad \quad \mathcal{Y} \quad \Leftarrow \quad \text{Tree Constraints} \]

\[ y(i,j) = 1 \quad \text{if} \quad y \text{ contains dependency } i, j \]

Penalties

\[ u(i,j) = 0 \quad \text{for all } i,j \]
**Individual Decoding**

\[ z^* = \arg \max_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

**Minimum Spanning Tree**

\[ y^* = \arg \max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

**Key**

- \( f(z) \Leftarrow \text{Sibling Model} \)
- \( g(y) \Leftarrow \text{Arc-Factored Model} \)
- \( \mathcal{Z} \Leftarrow \text{No Constraints} \)
- \( \mathcal{Y} \Leftarrow \text{Tree Constraints} \)
- \( y(i,j) = 1 \text{ if } y \text{ contains dependency } i,j \)
Individual Decoding

\[ z^* = \underset{z \in Z}{\text{arg max}} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

Minimum Spanning Tree

\[ y^* = \underset{y \in \mathcal{Y}}{\text{arg max}} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

Key

- \( f(z) \) \( \leftarrow \) Sibling Model
- \( Z \) \( \leftarrow \) No Constraints
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i,j \)
- \( g(y) \) \( \leftarrow \) Arc-Factored Model
- \( \mathcal{Y} \) \( \leftarrow \) Tree Constraints

Penalties

\[ u(i,j) = 0 \text{ for all } i,j \]
Individual Decoding

\[ z^* = \arg \max_{z \in Z} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

Penalties

\[ u(i,j) = 0 \text{ for all } i,j \]

Minimum Spanning Tree

\[ y^* = \arg \max_{y \in Y} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

Key

- \( f(z) \) ⇐ Sibling Model
- \( g(y) \) ⇐ Arc-Factored Model
- \( Z \) ⇐ No Constraints
- \( Y \) ⇐ Tree Constraints
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i,j \)
Individual Decoding

\[ z^* = \arg \max_{z \in Z} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

Minimum Spanning Tree

\[ y^* = \arg \max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

Penalties

\[
\begin{array}{c|c}
\text{Iteration 1} & \\
\hline
u(8,1) & -1 \\
u(4,6) & -1 \\
u(2,6) & 1 \\
u(8,7) & 1 \\
\end{array}
\]

Key

- \( f(z) \leftarrow \text{Sibling Model} \)
- \( g(y) \leftarrow \text{Arc-Factored Model} \)
- \( \mathcal{Z} \leftarrow \text{No Constraints} \)
- \( \mathcal{Y} \leftarrow \text{Tree Constraints} \)
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i,j \)
Individual Decoding

\[ z^* = \arg \max_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

Minimum Spanning Tree

\[ y^* = \arg \max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

Key

- \( f(z) \) ⇐ Sibling Model
- \( \mathcal{Z} \) ⇐ No Constraints
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i,j \)
- \( g(y) \) ⇐ Arc-Factored Model
- \( \mathcal{Y} \) ⇐ Tree Constraints

Penalties

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( u(8, 1) )</td>
<td>-1</td>
</tr>
<tr>
<td>( u(4, 6) )</td>
<td>-1</td>
</tr>
<tr>
<td>( u(2, 6) )</td>
<td>1</td>
</tr>
<tr>
<td>( u(8, 7) )</td>
<td>1</td>
</tr>
</tbody>
</table>
Individual Decoding

$z^* = \arg \max_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j)z(i,j))$

Minimum Spanning Tree

$y^* = \arg \max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j)y(i,j))$

Key

$f(z) \Leftarrow \text{Sibling Model}$
$\mathcal{Z} \Leftarrow \text{No Constraints}$
$y(i,j) = 1 \text{ if } y \text{ contains dependency } i,j$

$g(y) \Leftarrow \text{Arc-Factored Model}$
$\mathcal{Y} \Leftarrow \text{Tree Constraints}$

Penalties

<table>
<thead>
<tr>
<th>Iteration 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u(8, 1)$</td>
</tr>
<tr>
<td>$u(4, 6)$</td>
</tr>
<tr>
<td>$u(2, 6)$</td>
</tr>
<tr>
<td>$u(8, 7)$</td>
</tr>
</tbody>
</table>
Individual Decoding

\[ z^* = \arg \max_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

Minimum Spanning Tree

\[ y^* = \arg \max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

Key

- \( f(z) \) $\Leftarrow$ Sibling Model
- \( \mathcal{Z} \) $\Leftarrow$ No Constraints
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i,j \)
- \( g(y) \) $\Leftarrow$ Arc-Factored Model
- \( \mathcal{Y} \) $\Leftarrow$ Tree Constraints

Penalties

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th>Iteration 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u(8, 1) )</td>
<td>( u(8, 1) )</td>
</tr>
<tr>
<td>(-1)</td>
<td>(-1)</td>
</tr>
<tr>
<td>( u(4, 6) )</td>
<td>( u(4, 6) )</td>
</tr>
<tr>
<td>(-1)</td>
<td>(-2)</td>
</tr>
<tr>
<td>( u(2, 6) )</td>
<td>( u(2, 6) )</td>
</tr>
<tr>
<td>( 1)</td>
<td>( 2)</td>
</tr>
<tr>
<td>( u(8, 7) )</td>
<td>( u(8, 7) )</td>
</tr>
<tr>
<td>( 1)</td>
<td>( 1)</td>
</tr>
</tbody>
</table>
Individual Decoding

\[ z^* = \operatorname{arg \ max}_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

Minimum Spanning Tree

\[ y^* = \operatorname{arg \ max}_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

Key

- \( f(z) \) ⇒ Sibling Model
- \( g(y) \) ⇒ Arc-Factored Model
- \( \mathcal{Z} \) ⇒ No Constraints
- \( \mathcal{Y} \) ⇒ Tree Constraints
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i,j \)

Penalties

\[
\begin{array}{c|c}
\text{Iteration 1} & \text{U(i,j)} \\
\hline
u(8, 1) & -1 \\
u(4, 6) & -1 \\
u(2, 6) & 1 \\
u(8, 7) & 1 \\
\hline
\text{Iteration 2} & \text{U(i,j)} \\
\hline
u(8, 1) & -1 \\
u(4, 6) & -2 \\
u(2, 6) & 2 \\
u(8, 7) & 1 \\
\end{array}
\]

\( u(i,j) = 0 \) for all \( i,j \)
Individual Decoding

\[ z^* = \arg \max_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

Minimum Spanning Tree

\[ y^* = \arg \max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

Key

- \( f(z) \) \( \Leftarrow \) Sibling Model
- \( \mathcal{Z} \) \( \Leftarrow \) No Constraints
- \( g(y) \) \( \Leftarrow \) Arc-Factored Model
- \( \mathcal{Y} \) \( \Leftarrow \) Tree Constraints
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i,j \)

Penalties

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th>Iteration 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u(8,1) )</td>
<td>( u(8,1) )</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>( u(4,6) )</td>
<td>( u(4,6) )</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>( u(2,6) )</td>
<td>( u(2,6) )</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( u(8,7) )</td>
<td>( u(8,7) )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( u(i,j) = 0 \) for all \( i,j \)
Individual Decoding

\[ z^* = \arg \max_{z \in Z} (f(z) + \sum_{i,j} u(i,j)z(i,j)) \]

Minimum Spanning Tree

\[ y^* = \arg \max_{y \in Y} (g(y) - \sum_{i,j} u(i,j)y(i,j)) \]

Key

- \( f(z) \): Sibling Model
- \( Z \): No Constraints
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i,j \)
- \( g(y) \): Arc-Factored Model
- \( Y \): Tree Constraints

Penalties

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th>Iteration 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u(8, 1) )</td>
<td>( u(8, 1) )</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>( u(4, 6) )</td>
<td>( u(4, 6) )</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>( u(2, 6) )</td>
<td>( u(2, 6) )</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( u(8, 7) )</td>
<td>( u(8, 7) )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Individual Decoding

\[
z^* = \arg \max_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j)z(i,j))
\]

Minimum Spanning Tree

\[
y^* = \arg \max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j)y(i,j))
\]

Penalties

\[
\begin{align*}
\text{Iteration 1} & \\
\quad u(8, 1) & \quad -1 \\
\quad u(4, 6) & \quad -1 \\
\quad u(2, 6) & \quad 1 \\
\quad u(8, 7) & \quad 1 \\
\text{Iteration 2} & \\
\quad u(8, 1) & \quad -1 \\
\quad u(4, 6) & \quad -2 \\
\quad u(2, 6) & \quad 2 \\
\quad u(8, 7) & \quad 1 \\
\end{align*}
\]

Converged

\[
y^* = \arg \max_{y \in \mathcal{Y}} f(y) + g(y)
\]

Key

- \( f(z) \leftarrow \text{Sibling Model} \)
- \( g(y) \leftarrow \text{Arc-Factored Model} \)
- \( \mathcal{Z} \leftarrow \text{No Constraints} \)
- \( \mathcal{Y} \leftarrow \text{Tree Constraints} \)
- \( y(i,j) = 1 \text{ if } y \text{ contains dependency } i,j \)
Guarantees

Theorem

If at any iteration $y^{(k)} = z^{(k)}$, then $(y^{(k)}, z^{(k)})$ is the global optimum.

In experiments, we find the global optimum on 98% of examples.
Guarantees

Theorem
If at any iteration \( y(k) = z(k) \), then \((y(k), z(k))\) is the global optimum.

In experiments, we find the global optimum on 98% of examples.

If we do not converge to a match, we can still return an approximate solution (more in the paper).
Extensions

- Grandparent Models

\[ f(y) = \ldots + \text{score}(gp = ^0, head = \text{saw}_2, prev = \text{movie}_4, mod = \text{today}_5) \]

- Head Automata (Eisner, 2000)

  Generalization of Sibling models

  Allow arbitrary automata as local scoring function.
Experiments

Properties:
- Exactness
- Parsing Speed
- Parsing Accuracy
- Comparison to Individual Decoding
- Comparison to LP/ILP

Training:
- Averaged Perceptron (more details in paper)

Experiments on:
- CoNLL Datasets
- English Penn Treebank
- Czech Dependency Treebank
How often do we exactly solve the problem?

Percentage of examples where the dual decomposition finds an exact solution.
Parsing Speed

- Number of sentences parsed per second
- Comparable to dynamic programming for projective parsing
<table>
<thead>
<tr>
<th></th>
<th>Arc-Factored</th>
<th>Prev Best</th>
<th>Grandparent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan</td>
<td>89.7</td>
<td>91.5</td>
<td>91.8</td>
</tr>
<tr>
<td>Dut</td>
<td>82.3</td>
<td>85.6</td>
<td>85.8</td>
</tr>
<tr>
<td>Por</td>
<td>90.7</td>
<td>92.1</td>
<td>93.0</td>
</tr>
<tr>
<td>Slo</td>
<td>82.4</td>
<td>85.6</td>
<td>86.2</td>
</tr>
<tr>
<td>Swe</td>
<td>88.9</td>
<td>90.6</td>
<td>91.4</td>
</tr>
<tr>
<td>Tur</td>
<td>75.7</td>
<td>76.4</td>
<td>77.6</td>
</tr>
<tr>
<td>Eng</td>
<td>90.1</td>
<td>—</td>
<td>92.5</td>
</tr>
<tr>
<td>Cze</td>
<td>84.4</td>
<td>—</td>
<td>87.3</td>
</tr>
</tbody>
</table>

Prev Best - Best reported results for CoNLL-X data set, includes:
- Approximate search (McDonald and Pereira, 2006)
- Loop belief propagation (Smith and Eisner, 2008)
- (Integer) Linear Programming (Martins et al., 2009)
Comparison to Subproblems

\[ F_1 \text{ for dependency accuracy} \]
Comparison to LP/ILP

Martins et al. (2009): Proposes two representations of non-projective dependency parsing as a linear programming relaxation as well as an exact ILP.

- LP (1)
- LP (2)
- ILP

Use an LP/ILP Solver for decoding

We compare:
- Accuracy
- Exactness
- Speed

Both LP and dual decomposition methods use the same model, features, and weights $w$. 
Comparison to LP/ILP: Accuracy

All decoding methods have comparable accuracy
Comparison to LP/ILP: Exactness and Speed

Percentage with exact solution

Sentences per second
References I


References II


