Dual Decomposition and Linear Programming Relaxations for Inference in Natural Language Processing

Michael Collins

Joint work with Tommi Jaakkola, Terry Koo, Sasha Rush, and David Sontag
Inference: from Models to Predictions

Example problems in speech and NLP:
- Speech Recognition
- Sequence Modeling (e.g. extensions to HMM/CRF)
- Parsing
- Machine Translation

The decoding problem:
\[
y^* = \arg\max_y f(y)
\]

This talk: dual decomposition for decoding in NLP
Inference: from Models to Predictions

Example problems in speech and NLP:

- Speech Recognition
- Sequence Modeling (e.g. extensions to HMM/CRF)
- Parsing
- Machine Translation
Example problems in speech and NLP:

- Speech Recognition
- Sequence Modeling (e.g. extensions to HMM/CRF)
- Parsing
- Machine Translation

The decoding problem:

\[ y^* = \arg \max_y f(y) \]
Inference: from Models to Predictions

Example problems in speech and NLP:

- Speech Recognition
- Sequence Modeling (e.g. extensions to HMM/CRF)
- Parsing
- Machine Translation

The decoding problem:

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

This talk: dual decomposition for decoding in NLP
Non-Projective Dependency Parsing

Important problem in many languages.

Problem is **NP-Hard** for all but the simplest models.
Integration of a PCFG and an HMM

- Classical problem in NLP.
- The dynamic programming intersection is prohibitively slow and complicated to implement.
The MAP Problem in Markov Random Fields

We have binary variables $x_1 \ldots x_n$

The 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” associated with variables $x_i, x_j$
The MAP problem: \[ \arg \max_{x_1 \ldots x_n} \sum_{(i,j) \in E} f_{i,j}(x_i, x_j) \]
The MAP problem: \( \arg \max_{x_1 \ldots x_n} \sum_{(i,j) \in E} f_{i,j}(x_i, x_j) \)

An equivalent problem:

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

such that \( x_i = y_i \) for \( i = 1 \ldots n \)
The MAP problem: \[ \arg \max_{x_1 \ldots x_n} \sum_{(i,j) \in E} f_{i,j}(x_i, x_j) \]

An equivalent problem:

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

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

The Lagrangian:

\[
\sum_{(i,j) \in T_1} f_{i,j}(x_i, x_j) + \sum_{(i,j) \in T_2} f_{i,j}(y_i, y_j) + \sum_i u_i(x_i - y_i)
\]
Roadmap

Algorithm

Experiments

Other Examples
Non-Projective Dependency Parsing

- Starts at the root symbol *
- Each word has a exactly one parent word
- Produces a tree structure (no cycles)
- Dependencies can cross
Algorithm Outline

Arc-Factored Model

Sibling Model
Algorithm Outline

Arc-Factored Model

Dual Decomposition

Sibling Model
Arc-Factored

\[
f(y) =
\]

\[
\text{score}(\ast_0, \text{saw}_2) = \log p(\text{saw}_2 | \ast_0) \quad \text{(generative model)}
\]

\[
\text{score}(\ast_0, \text{saw}_2) = w \cdot \phi(\text{saw}_2, \ast_0) \quad \text{(CRF/perceptron model)}
\]

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

\[ f(y) = \text{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) \]
Arc-Factored

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

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

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

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

\[ f(y) = \text{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) + \text{score}(\text{saw}_2, \text{John}_1) \]
\[ + \text{score}(\text{saw}_2, \text{movie}_4) + \text{score}(\text{saw}_2, \text{today}_5) \]
\[ + \text{score}(\text{movie}_4, a_3) + \ldots \]
Arc-Factored

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

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

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

e.g. \( \text{score}(*_0, \text{saw}_2) = \log p(\text{saw}_2|*_0) \) (generative model) \\
\text{or } \text{score}(*_0, \text{saw}_2) = w \cdot \phi(\text{saw}_2, *_0) \) (CRF/perceptron 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.
\[ \text{score}(\ast_0, \text{saw}_2) = \log p(\text{saw}_2 | \ast_0) \] (generative model)

or
\[ \text{score}(\ast_0, \text{saw}_2) = w \cdot \phi(\text{saw}_2, \ast_0) \] (CRF/perceptron model)

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

\[ f(y) = \]

\[
\begin{align*}
*_{0} \quad & \text{John}_{1} \quad & \text{saw}_{2} \quad & a_{3} \quad & \text{movie}_{4} \quad & \text{today}_{5} \quad & \text{that}_{6} \quad & \text{he}_{7} \quad & \text{liked}_{8} \\
\end{align*}
\]

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

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

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

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

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

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

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

\[ \Leftarrow \text{NP-Hard} \]
Sibling Models

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

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

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

\[ f(y) = score(head = *_0, prev = NULL, mod = saw_2) \]
\[ + score(saw_2, NULL, John_1) + score(saw_2, NULL, movie_4) \]
\[ + score(saw_2, movie_4, today_5) + \ldots \]

e.g. \( score(saw_2, movie_4, today_5) = \log p(today_5|saw_2, movie_4) \)
Sibling Models

\[ f(y) = score(head = *_0, prev = NULL, mod = saw_2) \]
\[ + score(saw_2, NULL, John_1) + score(saw_2, NULL, movie_4) \]
\[ + score(saw_2, movie_4, today_5) + ... \]

e.g. \( score(saw_2, movie_4, today_5) = \log p(today_5 | saw_2, movie_4) \)

or \( score(saw_2, movie_4, today_5) = w \cdot \phi(saw_2, movie_4, today_5) \)
Sibling Models

\[ f(y) = score(head = *_0, prev = \text{NULL}, mod = \text{saw}_2) \]
\[ + score(saw_2, \text{NULL}, \text{John}_1) + score(saw_2, \text{NULL}, \text{movie}_4) \]
\[ + score(saw_2, \text{movie}_4, \text{today}_5) + \ldots \]

e.g. \( score(saw_2, \text{movie}_4, \text{today}_5) = \log p(\text{today}_5|\text{saw}_2, \text{movie}_4) \)

or \( score(saw_2, \text{movie}_4, \text{today}_5) = w \cdot \phi(saw_2, \text{movie}_4, \text{today}_5) \)

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

*0  John1  saw2  a3  movie4  today5  that6  he7  liked8
Thought Experiment: Individual Decoding

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

\[ \text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) \]
\[ + \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) \]
\[ + \text{score}(\text{saw}_2, \text{NULL}, \text{that}_6) \]
**Thought Experiment: Individual Decoding**

\[
\begin{align*}
&\text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) \\
&+ \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) \\
&\quad + \text{score}(\text{saw}_2, \text{NULL}, \text{that}_6) \\
&\quad + \text{score}(\text{saw}_2, \text{a}_3, \text{he}_7) \\
\end{align*}
\]
Thought Experiment: 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}

2^{n-1} \text{ possibilities}

\begin{align*}
\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)
\end{align*}

\begin{align*}
\text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \text{score}(\text{saw}_2, \text{NULL}, \text{that}_6)
\end{align*}

\begin{align*}
\text{score}(\text{saw}_2, \text{NULL}, \text{a}_3) + \text{score}(\text{saw}_2, \text{a}_3, \text{he}_7)
\end{align*}
Thought Experiment: Individual Decoding

*0 John1 saw2 a3 movie4 today5 that6 he7 liked8

\[
\begin{align*}
\text{score}(\text{saw}_2, \text{NULL}, \text{John}_1) &+ \text{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) \\
+ \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) &+ \text{score}(\text{saw}_2, \text{NULL}, \text{that}_6) \\
\text{score}(\text{saw}_2, \text{NULL}, \text{a}_3) &+ \text{score}(\text{saw}_2, \text{a}_3, \text{he}_7)
\end{align*}
\]

Under Sibling Model, can solve for each word with Viterbi decoding.
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.
Dual Decomposition Idea

- Arc-Factored
  - No Constraints
  - Sibling Model
- Tree Constraints
  - Minimum Spanning Tree
  - Individual Decoding
Dual Decomposition Idea

<table>
<thead>
<tr>
<th></th>
<th>No Constraints</th>
<th>Tree Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arc-Factored Sibling Model</td>
<td></td>
<td>Minimum Spanning Tree</td>
</tr>
<tr>
<td>Individual Decoding</td>
<td></td>
<td>Dual Decomposition</td>
</tr>
</tbody>
</table>
Dual Decomposition Structure

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

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

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 \( z = y \)
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

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 $z = y$
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 $z = y$
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 $z = y$
Algorithm Sketch

Set penalty weights equal to 0 for all edges.

For $k = 1$ to $K$
Algorithm Sketch

Set penalty weights equal to 0 for all edges.

For $k = 1$ to $K$

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

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

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

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

Else Update penalty weights based on $y^{(k)}(i, j) - z^{(k)}(i, j)$
**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
- \( g(y) \) ⇐ Arc-Factored Model
- \( \mathcal{Z} \) ⇐ No Constraints
- \( \mathcal{Y} \) ⇐ Tree Constraints
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i, j \)

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

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

\( \mathcal{Z} \Leftarrow \text{No Constraints} \quad \mathcal{Y} \Leftarrow \text{Tree Constraints} \)

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

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

Minimum Spanning Tree

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

Key

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

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
- \( y(i,j) = 1 \text{ if } y \text{ contains dependency } i,j \)
Individual Decoding

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

Minimum Spanning Tree

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

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
- \( Y \) \( \Leftarrow \) Tree Constraints

Penalties

\[
\begin{array}{c|c}
\text{Iteration 1} & \\
\hline
u(8, 1) & -1 \\
u(4, 6) & -1 \\
u(2, 6) & 1 \\
u(8, 7) & 1 \\
\end{array}
\]
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
- \(g(y)\) ← Arc-Factored Model
- \(\mathcal{Z}\) ← No Constraints
- \(\mathcal{Y}\) ← Tree Constraints
- \(y(i,j) = 1\) if \(y\) contains dependency \(i,j\)

Penalties

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

\(u(i, j) = 0\) 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
- \( g(y) \) \( \Leftarrow \) Arc-Factored Model
- \( \mathcal{Z} \) \( \Leftarrow \) No Constraints
- \( \mathcal{Y} \) \( \Leftarrow \) Tree Constraints
- \( y(i,j) = 1 \) if \( y \) contains dependency \( i,j \)

Penalties

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

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

Key

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

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

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

Penalties

\[
\begin{array}{c|c}
\text{Iteration 1} & \\
\hline
u(8,1) & -1 \\
u(4,6) & -1 \\
u(2,6) & 1 \\
u(8,7) & 1 \\
\hline
\text{Iteration 2} & \\
\hline
u(8,1) & -1 \\
u(4,6) & -2 \\
u(2,6) & 2 \\
u(8,7) & 1 \\
\end{array}
\]
**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
- **g(y)** ← Arc-Factored Model
- **Z** ← No Constraints
- **Y** ← 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>-1</td>
</tr>
<tr>
<td>u(4, 6)</td>
<td>-2</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

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

Minimum Spanning Tree

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

Penalties

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

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

Converged

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

Key

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

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

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

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

Rewrite:
\[ \arg \max_{z \in \mathcal{Z}, y \in \mathcal{Y}} f(z) + g(y) \]
\[ \text{s.t. } z(i, j) = y(i, j) \text{ for all } i, j \]

Lagrangian: \[ L(u, y, z) = f(z) + g(y) + \sum_{i,j} u(i, j) (z(i, j) - y(i, j)) \]
Deriving the Algorithm

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

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

s.t. \( z(i, j) = y(i, j) \) for all \( i, j \)

Lagrangian: \( L(u, y, z) = f(z) + g(y) + \sum_{i,j} u(i, j) (z(i, j) - y(i, j)) \)

The dual problem is to find \( \min_u L(u) \) where

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

\[
+ \max_{y \in \mathcal{Y}} \left( g(y) - \sum_{i,j} u(i, j) y(i, j) \right) \]

Dual is an upper bound: \( L(u) \geq f(z^*) + g(y^*) \) for any \( u \)
A Subgradient Algorithm for Minimizing $L(u)$

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

$L(u)$ is convex, but not differentiable. 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 methods use updates $u' = u - \alpha g_u$

In fact, for our $L(u)$, $g_u(i,j) = z^*(i,j) - y^*(i,j)$
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.
<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

![Graph showing F1 for dependency accuracy for Individual, MST, and Dual methods.]

- **Individual**
- **MST**
- **Dual**

F1 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
Roadmap

Algorithm

Experiments

Other Examples
Integrated Parsing and Tagging

Red₁ flies₂ some₃ large₄ jet₅

N → V → D → A → N

Red₁ flies₂ some₃ large₄ jet₅

Red₁ flies₂ some₃ large₄ jet₅

NP S
N V NP
Red flies D A N
some large jet
Integrated Parsing and Tagging

Red_1 flies_2 some_3 large_4 jet_5

N → V → D → A → N

Red_1 flies_2 some_3 large_4 jet_5

Red_1 flies_2 some_3 large_4 jet_5

S

NP

V

NP

Red

flies

d some

large

jet

HMM

Dual Decomposition

CFG
Let $Z$ be the set of all valid taggings of a sentence and $g(z)$ be a scoring function.

E.g. $g(z) = \log p(\text{Red}_1 | N) + \log p(V | N) + ...$
Let $Z$ be the set of all valid taggings of a sentence and $g(z)$ be a scoring function.

\[ g(z) = \log p(\text{Red}_1|N) + \log p(V|N) + \ldots \]

\[ z^* = \arg \max_{z \in Z} g(z) \leftarrow \text{Viterbi decoding} \]
Let $\mathcal{Y}$ be the set of all valid parse trees for a sentence and $f(y)$ be a scoring function.

\[
e.g. \quad f(y) = \log p(S \rightarrow NP \ VP | S) + \log p(NP \rightarrow N|NP) + \ldots
\]
Let $\mathcal{Y}$ be the set of all valid parse trees for a sentence and $f(y)$ be a scoring function.

e.g. $f(y) = \log p(S \rightarrow NP \ VP|S) + \log p(NP \rightarrow N|NP) + ...$

$y^* = \arg \max_{y \in \mathcal{Y}} f(y) \leftarrow \text{CKY Algorithm}$
Problem Definition

Find parse tree that optimizes

\[ \text{score}(S \rightarrow NP \ VP) + \text{score}(VP \rightarrow V \ NP) + \]
\[ \ldots + \text{score}(\text{Red}_1, N) + \text{score}(V, N) + \ldots \]

Conventional Approach (Bar Hillel et al., 1961)

- Replace rules like \( S \rightarrow NP \ VP \)
  - with rules like \( S_{N,N} \rightarrow NP_{N,V} \ VP_{V,N} \)

Painful. \( O(t^6) \) increase in complexity for trigram tagging.
The Integrated Parsing and Tagging Problem

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

such that for all \( i, t, \ y(i, t) = z(i, t) \)

Where \( y(i, t) = 1 \) if parse includes tag \( t \) at position \( i \)
\( z(i, t) = 1 \) if tagging includes tag \( t \) at position \( i \)
The Integrated Parsing and Tagging Problem

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

such that for all $i, t$, $y(i, t) = z(i, t)$

Where $y(i, t) = 1$ if parse includes tag $t$ at position $i$

$z(i, t) = 1$ if tagging includes tag $t$ at position $i$
The Integrated Parsing and Tagging Problem

Find \( \arg\max \ y \in \mathcal{Y}, \ z \in \mathcal{Z} \ f(y) + g(z) \)

such that for all \( i, t \), \( y(i, t) = z(i, t) \)

Where \( y(i, t) = 1 \) if parse includes tag \( t \) at position \( i \)

\( z(i, t) = 1 \) if tagging includes tag \( t \) at position \( i \)
The Integrated Parsing and Tagging Problem

Find \( \text{argmax} \ y \in Y, z \in Z \ f(y) + g(z) \)

such that for all \( i, t, y(i, t) = z(i, t) \)

Where \( y(i, t) = 1 \) if parse includes tag \( t \) at position \( i \)

\( z(i, t) = 1 \) if tagging includes tag \( t \) at position \( i \)
The Integrated Parsing and Tagging Problem

Find \( \text{argmax} \ f(y) + g(z) \)

\( y \in \mathcal{Y}, z \in \mathcal{Z} \)

such that for all \( i, t, y(i, t) = z(i, t) \)

Where \( y(i, t) = 1 \) if parse includes tag \( t \) at position \( i \)

\( z(i, t) = 1 \) if tagging includes tag \( t \) at position \( i \)
The Integrated Parsing and Tagging Problem

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

such that for all \( i, t \), \( y(i, t) = z(i, t) \)

Where \( y(i, t) = 1 \) if parse includes tag \( t \) at position \( i \)

\( z(i, t) = 1 \) if tagging includes tag \( t \) 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

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

Penalties

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

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 \( i \)
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

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

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

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

\[ S \]

\[ NP \quad VP \]

\[ A \quad N \quad D \quad A \quad V \]

Red flies some large jet

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

Viterbi Decoding

\[ N \rightarrow V \rightarrow D \rightarrow A \rightarrow N \]

\[ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \]

\[ \text{Red}_1 \quad \text{flies}_2 \quad \text{some}_3 \quad \text{large}_4 \quad \text{jet}_5 \]

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

Key

\[ f(y) \quad \Leftarrow \quad \text{CFG} \]

\[ g(z) \quad \Leftarrow \quad \text{HMM} \]

\[ Y \quad \Leftarrow \quad \text{Parse Trees} \]

\[ Z \quad \Leftarrow \quad \text{Taggings} \]

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

NP  
Red  flies  some  large  jet

VP

\[ S \]

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

Viterbi Decoding

N  \rightarrow  V  \rightarrow  D  \rightarrow  A  \rightarrow  N

\[ \downarrow \]  \[ \downarrow \]  \[ \downarrow \]  \[ \downarrow \]  \[ \downarrow \]

Red_1  flies_2  some_3  large_4  jet_5

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

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>

Key

<table>
<thead>
<tr>
<th>Function</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(y) )</td>
<td>CFG</td>
</tr>
<tr>
<td>( g(z) )</td>
<td>HMM</td>
</tr>
<tr>
<td>( y )</td>
<td>Parse Trees</td>
</tr>
<tr>
<td>( z )</td>
<td>Taggings</td>
</tr>
<tr>
<td>( y(i, t) = 1 )</td>
<td>y contains tag t at position i</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

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

Penalties

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

\begin{align*}
\text{Iteration 1} & \\
& \begin{array}{c|c}
(i, t) & u(i, t) \\
\hline
(1, A) & -1 \\
(1, N) & 1 \\
(2, N) & -1 \\
(2, V) & 1 \\
(5, V) & -1 \\
(5, N) & 1 \\
\end{array}
\end{align*}

Key

\[ f(y) \Leftarrow \text{CFG} \quad \quad g(z) \Leftarrow \text{HMM} \]

\[ Y \Leftarrow \text{Parse Trees} \quad \quad Z \Leftarrow \text{Taggings} \]

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

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

Viterbi Decoding

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

Penalties

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

<table>
<thead>
<tr>
<th></th>
<th>Iteration 1</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>

Key

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

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

\[ S \]
\[ \text{NP} \quad \text{VP} \]
\[ \text{N} \quad \text{V} \]
\[ \text{Red} \quad \text{flies} \quad \text{D} \quad \text{A} \quad \text{N} \]
\[ \text{some} \quad \text{large} \quad \text{jet} \]

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

Viterbi Decoding

\[ \text{A} \rightarrow \text{N} \rightarrow \text{D} \rightarrow \text{A} \rightarrow \text{N} \]
\[ \text{Red}_1 \quad \text{flies}_2 \quad \text{some}_3 \quad \text{large}_4 \quad \text{jet}_5 \]

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

Penalties

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

Iteration 1

\[
\begin{array}{c|c}
\text{i} & \text{t} \\
\hline
A & -1 \\
N & 1 \\
D & -1 \\
A & 1 \\
N & 1 \\
\end{array}
\]

Iteration 2

\[
\begin{array}{c|c}
\text{i} & \text{t} \\
\hline
5 & V & -1 \\
5 & N & 1 \\
\end{array}
\]

Key

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

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

Viterbi Decoding

Red_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

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

Penalties

\[
\begin{array}{l l}
\text{Iteration 1} & \\
   & \\
   & \hline
   \mathcal{A} & -1 \\
   \mathcal{N} & 1 \\
   \mathcal{V} & -1 \\
   \mathcal{N} & 1 \\
\end{array}
\]

\[
\begin{array}{l l}
\text{Iteration 2} & \\
   & \\
   & \hline
   \mathcal{V} & -1 \\
   \mathcal{N} & 1 \\
\end{array}
\]

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

- \( f(y) \) \( \Leftarrow \) CFG
- \( g(z) \) \( \Leftarrow \) HMM
- \( \mathcal{Y} \) \( \Leftarrow \) Parse Trees
- \( \mathcal{Z} \) \( \Leftarrow \) 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

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

Iteration 2

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

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

Viterbi Decoding

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

Key

\[
f(y) \Leftarrow \text{CFG} \quad g(z) \Leftarrow \text{HMM}
\]

\[
\mathcal{Y} \Leftarrow \text{Parse Trees} \quad \mathcal{Z} \Leftarrow \text{Taggings}
\]

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

Penalties

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

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

Iteration 1

\[
u(1, A) = -1
\]

\[
u(1, N) = 1
\]

\[
u(2, N) = -1
\]

\[
u(2, V) = 1
\]

\[
u(5, V) = -1
\]

\[
u(5, N) = 1
\]

Iteration 2

\[
u(5, V) = -1
\]

\[
u(5, N) = 1
\]

Converged

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

Properties:

- Exactness
- Parsing Accuracy

Experiments on:

- English Penn Treebank

Models

- Collins (1997) Model 1
- Semi-Supervised Dependency Parser (Koo, 2008)
- Trigram Tagger (Toutanova, 2000)
How quickly do the models converge?

Integrated Dependency Parsing

Integrated POS Tagging
Integrated Constituency and Dependency Parsing: Accuracy

F₁ Score
- Collins (1997) Model 1
- Fixed, First-best Dependencies from Koo (2008)
- Dual Decomposition
F_1 Score

- Fixed, First-Best Tags From Toutanova (2000)
- Dual Decomposition
Future Directions

There is much more to explore around dual decomposition in NLP.

- **Known Techniques**
  - Generalization to more than two models
  - K-best decoding
  - Approximate subgradient
  - Heuristic for branch-and-bound type search

- **Possible NLP Applications**
  - Machine Translation
  - Speech Recognition
  - “Loopy” Sequence Models

- **Open Questions**
  - Can we speed up subalgorithms when running repeatedly?
  - What are the trade-offs of different decompositions?
  - Are there better methods for optimizing the dual?
Roadmap

Algorithm

Experiments

LP Relaxations
Dual Decomposition and Linear Programming Relaxations

Theorem
- If the dual decomposition algorithm converges, then \((y^{(k)}, z^{(k)})\) is the global optimum.

Questions
- What problem is dual decomposition solving?
- How come the algorithm doesn't always converge?

Dual decomposition searches over a linear programming relaxation of the original problem.
Convex Hulls for CKY

A parse tree can be represented as a binary vector $y \in \mathcal{Y}$. $y(A \rightarrow B \ C, i, j, k) = 1$ if rule $A \rightarrow B \ C$ is used at span $i, j, k$.

- If $f$ is linear, $\arg \max_{y \in \text{conv}(\mathcal{Y})} f(y)$ is a linear program.
- The best point in an LP is a vertex. So CKY solves this LP.
A parse tree can be represented as a binary vector $y \in \mathcal{Y}$. $y(A \to B \ C, i, j, k) = 1$ if rule $A \to B \ C$ is used at span $i, j, k$.

- If $f$ is linear, $\arg \max_{y \in \text{conv}(\mathcal{Y})} f(y)$ is a linear program.
- The best point in an LP is a vertex. So CKY solves this LP.
A parse tree can be represented as a binary vector $y \in \mathcal{Y}$. $y(A \rightarrow B C, i, j, k) = 1$ if rule $A \rightarrow B C$ is used at span $i, j, k$.

If $f$ is linear, $\arg \max_{y \in \text{conv} \mathcal{Y}} f(y)$ is a linear program.

The best point in an LP is a vertex. So CKY solves this LP.
Convex Hulls for CKY

A parse tree can be represented as a binary vector \( y \in \mathcal{Y} \).
\( y(A \to B \ B \ C, i, j, k) = 1 \) if rule \( A \to B \ B \ C \) is used at span \( i, j, k \).

If \( f \) is linear, \( \arg \max_{y \in \text{conv}(\mathcal{Y})} f(y) \) is a linear program.

The best point in an LP is a vertex. So CKY solves this LP.
Combined Problem

\[ Q = \{(y, z): y \in \mathcal{Y}, z \in \mathcal{Z}, \]
\[ y(i, t) = z(i, t) \text{ for all } (i, t)\}\]
Combined Problem

\[ Q = \{(y, z): y \in \mathcal{Y}, z \in \mathcal{Z},\]
\[ y(i, t) = z(i, t) \text{ for all } (i, t)\}\]
Combined Problem

\[ Q = \{ (y, z) : y \in \mathcal{Y}, z \in \mathcal{Z}, \\
\quad y(i, t) = z(i, t) \text{ for all } (i, t) \} \]

\[ \text{conv}(Q) \]
Combined Problem

\[ Q = \{(y, z): y \in \mathcal{Y}, z \in \mathcal{Z}, \]
\[ y(i, t) = z(i, t) \text{ for all } (i, t)\}\]

\[ Q' = \{(\mu, \nu): \mu \in \text{conv}(\mathcal{Y}), \nu \in \text{conv}(\mathcal{Z}), \]
\[ \mu(i, t) = \nu(i, t) \text{ for all } (i, t)\}\]

Dual decomposition searches over \(Q'\)
Combined Problem

\[ Q = \{ (y, z): y \in \mathcal{Y}, z \in \mathcal{Z}, \]
\[ y(i, t) = z(i, t) \text{ for all } (i, t) \} \]

\[ Q' = \{ (\mu, \nu): \mu \in \text{conv}(\mathcal{Y}), \nu \in \text{conv}(\mathcal{Z}), \]
\[ \mu(i, t) = \nu(i, t) \text{ for all } (i, t) \} \]

Dual decomposition searches over \( Q' \)
Combined Problem

\[ Q = \{ (y, z): y \in \mathcal{Y}, z \in \mathcal{Z}, \quad y(i, t) = z(i, t) \text{ for all } (i, t) \} \]

\[ Q' = \{ (\mu, \nu): \mu \in \text{conv}(\mathcal{Y}), \nu \in \text{conv}(\mathcal{Z}), \quad \mu(i, t) = \nu(i, t) \text{ for all } (i, t) \} \]

Dual decomposition searches over \( Q' \)

Depending on the weight vector, \((y^*, z^*) \in Q'\) could be in \( Q \) or in the strict outer bound.
Combined Problem

\[ Q = \{(y, z): y \in \mathcal{Y}, z \in \mathcal{Z}, \quad y(i, t) = z(i, t) \text{ for all } (i, t)\} \]

\[ Q' = \{(\mu, \nu): \mu \in \text{conv}(\mathcal{Y}), \nu \in \text{conv}(\mathcal{Z}), \quad \mu(i, t) = \nu(i, t) \text{ for all } (i, t)\} \]

Dual decomposition searches over \( Q' \)

Depending on the weight vector, \((y^*, z^*) \in Q'\) could be in \( Q \) or in the strict outer bound.
Are there points strictly in the outer bound?

\[ Q' \]

Possible \((y^*, z^*)\)?

Taggings

\[
\begin{align*}
0.5x & \quad A \longrightarrow A \longrightarrow A \\
& \quad w_1 \quad w_2 \quad w_3
\end{align*}
\]

\[
\begin{align*}
+ 0.5x & \quad A \longrightarrow B \longrightarrow B \\
& \quad w_1 \quad w_2 \quad w_3
\end{align*}
\]

Best result can be a fractional solution.

Parses

\[
\begin{align*}
0.5x & \quad X \quad \frac{A}{A} \quad X \\
& \quad w_1 \quad w_2 \quad w_3
\end{align*}
\]

\[
\begin{align*}
+ 0.5x & \quad X \quad \frac{A}{A} \quad X \\
& \quad w_1 \quad w_2 \quad w_3
\end{align*}
\]

Convex combination of these structures.
A Dual Decomposition algorithm for integrated decoding

**Simple** - Uses only simple, off-the-shelf dynamic programming algorithms to solve a harder problem.

**Efficient** - Faster than classical methods for dynamic programming intersection.

**Strong Guarantees** - Solves a linear programming relaxation which gives a certificate of optimality.

Finds the exact solution on 99% of the examples.

**Widely Applicable** - Similar techniques extend to other problems
Appendix
Deriving the Algorithm

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

Rewrite:
\[ \arg \max_{z \in \mathcal{Z}, y \in \mathcal{Y}} f(z) + g(y) \]
\[ \text{s.t. } z(i, j) = y(i, j) \text{ for all } i, j \]

Lagrangian: \[ L(u, y, z) = f(z) + g(y) + \sum_{i,j} u(i, j) (y(i, j) - z(i, j)) \]
Deriving the Algorithm

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

Rewrite:
\[ \arg \max_{z \in \mathcal{Z}, y \in \mathcal{Y}} f(z) + g(y) \]
\[ \text{s.t. } z(i, j) = y(i, j) \text{ for all } i, j \]

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

The dual problem is to find \( \min_u L(u) \) where

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

Dual is an upper bound: \( L(u) \geq f(z^*) + g(y^*) \) for any \( u \)
A Subgradient Algorithm for Minimizing $L(u)$

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

$L(u)$ is convex, but not differentiable. 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 methods use updates $u' = u - \alpha g_u$

In fact, for our $L(u)$, $g_u(i,j) = z^*(i,j) - y^*(i,j)$
Related Work

- Methods that use general purpose linear programming or integer linear programming solvers (Martins et al. 2009; Riedel and Clarke 2006; Roth and Yih 2005)


- Dual decomposition for inference in MRFs (Komodakis et al., 2007; Wainwright et al., 2005)

- Methods that incorporate combinatorial solvers within loopy belief propagation (Duchi et al. 2007; Smith and Eisner 2008)
Related Work

- Methods that use general purpose linear programming or integer linear programming solvers (Martins et al. 2009; Riedel and Clarke 2006; Roth and Yih 2005)
- Dual decomposition for inference in MRFs (Komodakis et al., 2007; Wainwright et al., 2005)
- Methods that incorporate combinatorial solvers within loopy belief propagation (Duchi et al. 2007; Smith and Eisner 2008)
Summary

\[ y^* = \arg \max_y \ f(y) \leq \text{NP-Hard} \]

Arc-Factored Model

Sibling Model
Summary

\[ y^* = \arg \max_y f(y) \leq \text{NP-Hard} \]

Arc-Factored Model

Dual Decomposition

Sibling Model
Other Applications

- Dual decomposition can be applied to other decoding problems.

- Rush et al. (2010) focuses on integrated dynamic programming algorithms.
  - Integrated Parsing and Tagging
  - Integrated Constituency and Dependency Parsing
Parsing and Tagging

\[ y^* = \arg \max_y f(y) \Leftarrow \text{Slow} \]

\[ S \rightarrow N \rightarrow V \rightarrow D \rightarrow N \rightarrow A \rightarrow D \rightarrow N \rightarrow V \]

HMM Model

\[ S \rightarrow N \rightarrow V \rightarrow D \rightarrow N \rightarrow A \rightarrow D \rightarrow N \rightarrow V \]

CFG Model
**Parsing and Tagging**

\[ y^* = \arg \max_y f(y) \Leftarrow \text{Slow} \]

---

**HMM Model**

**Dual Decomposition**

**CFG Model**
Dependency and Constituency

\[ y^* = \arg \max_y f(y) \leftarrow \text{Slow} \]

Dependency Model

Lexicalized CFG
Dependency and Constituency

\[ y^* = \arg \max_y f(y) \iff \text{Slow} \]

*0  John₁  saw₂  a₃  movie₄  today₅  that₆  he₇  liked₈

**Dependency Model**

**Dual Decomposition**

**Lexicalized CFG**
A Dual Decomposition Algorithm for Non-Projective Dependency Parsing

Simple - Uses basic combinatorial algorithms

Efficient - Faster than previously proposed algorithms

Strong Guarantees - Gives a certificate of optimality when exact

Solves 98% of examples exactly, even though the problem is NP-Hard

Widely Applicable - Similar techniques extend to other problems
Future Directions

There is much more to explore around dual decomposition in NLP.

- **Known Techniques**
  - Generalization to more than two models
  - K-best decoding
  - Approximate subgradient
  - Heuristic for branch-and-bound type search

- **Possible NLP Applications**
  - Machine Translation
  - Speech Recognition
  - “Loopy” Sequence Models

- **Open Questions**
  - Can we speed up subalgorithms when running repeatedly?
  - What are the trade-offs of different decompositions?
  - Are there better methods for optimizing the dual?
Appendix
Training the Model

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

- \( \text{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) = w \cdot \phi(\text{saw}_2, \text{movie}_4, \text{today}_5) \)
- Weight vector \( w \) trained using Averaged perceptron.
- (More details in the paper.)
Early Stopping

![Graph showing Early Stopping]

- % validation UAS
- % certificates
- % match K=5000

Maximum Number of Dual Decomposition Iterations vs. Percentage
Caching speed