Dual Decomposition for Parsing with Non-Projective Head Automata

Terry Koo, Alexander M. Rush, Michael Collins,
David Sontag, and Tommi Jaakkola

The Cost of Model Complexity

We are always looking for better ways to model natural language.

Tradeoff: Richer models $\Rightarrow$ Harder decoding

Added complexity is both computational and implementational.

The Cost of Model Complexity

We are always looking for better ways to model natural language.

Tradeoff: Richer models $\Rightarrow$ Harder decoding

Added complexity is both computational and implementational.

Tasks with challenging decoding problems:

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

The Cost of Model Complexity

We are always looking for better ways to model natural language.

Tradeoff: Richer models $\Rightarrow$ Harder decoding

Added complexity is both computational and implementational.

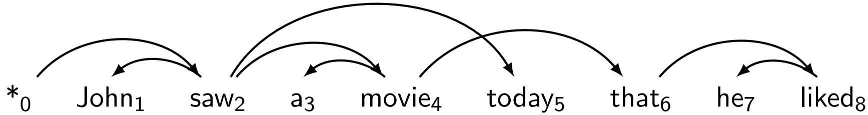
Tasks with challenging decoding problems:

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

$$y^* = \arg\max_y f(y) \quad \text{Decoding}$$

Non-Projective Dependency Parsing

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$



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
- ...

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

Roadmap

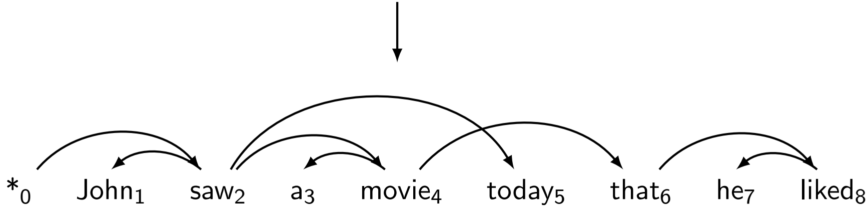
Algorithm

Experiments

Derivation

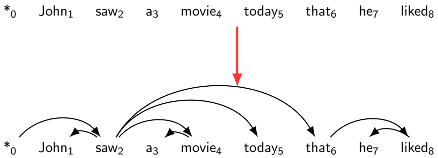
Non-Projective Dependency Parsing

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

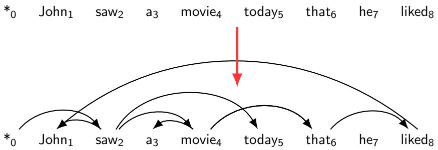


- Starts at the root symbol *
- Each word has a exactly one parent word
- Produces a tree structure (no cycles)
- Dependencies can cross

Algorithm Outline

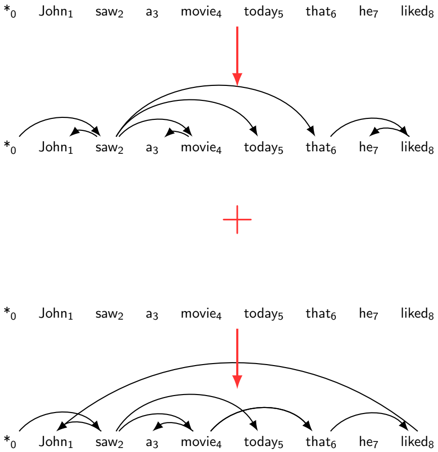


Arc-Factored Model



Sibling Model

Algorithm Outline



Arc-Factored Model

Dual Decomposition

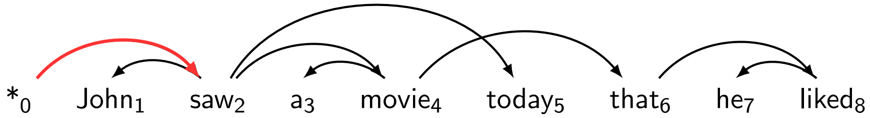
Sibling Model

Arc-Factored

$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

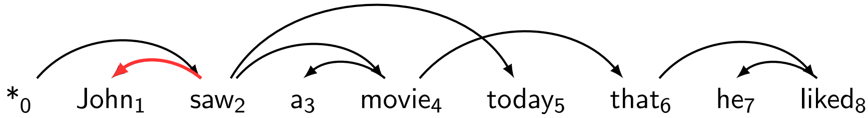
$f(y) =$

Arc-Factored



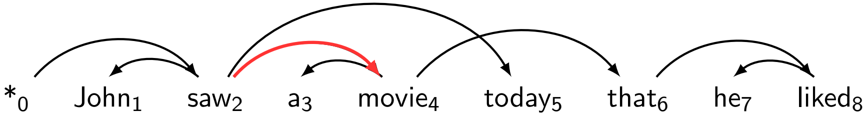
$$f(y) = \mathit{score}(\mathrm{head} = *_0, \mathrm{mod} = \mathrm{saw}_2)$$

Arc-Factored



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

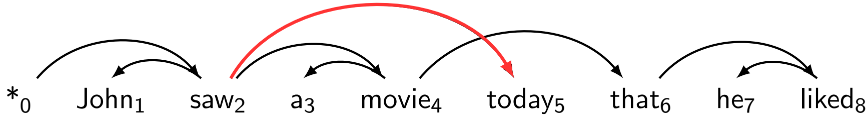
Arc-Factored



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

$$+\textit{score}(\text{saw}_2, \text{movie}_4)$$

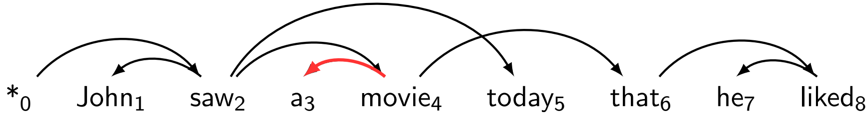
Arc-Factored



$$f(y) = \mathit{score}(\mathrm{head} = *_0, \mathrm{mod} = \mathrm{saw}_2) + \mathit{score}(\mathrm{saw}_2, \mathrm{John}_1)$$

$$+ \mathit{score}(\mathrm{saw}_2, \mathrm{movie}_4) + \mathit{score}(\mathrm{saw}_2, \mathrm{today}_5)$$

Arc-Factored

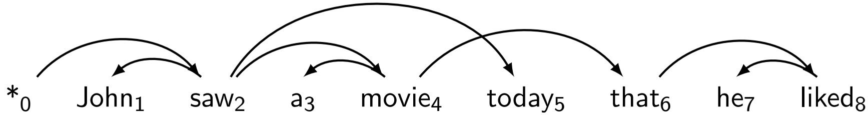


$$f(y) = \mathit{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) + \mathit{score}(\text{saw}_2, \text{John}_1)$$

$$+\mathit{score}(\text{saw}_2, \text{movie}_4) + \mathit{score}(\text{saw}_2, \text{today}_5)$$

$$+\mathit{score}(\text{movie}_4, \text{a}_3) + \ldots$$

Arc-Factored



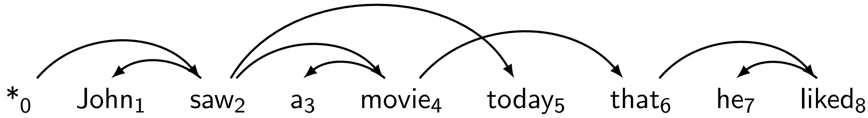
$$f(y) = \textit{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) + \textit{score}(\text{saw}_2, \text{John}_1)$$

$$+\textit{score}(\text{saw}_2, \text{movie}_4) + \textit{score}(\text{saw}_2, \text{today}_5)$$

$$+\textit{score}(\text{movie}_4, \text{a}_3) + \ldots$$

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

Arc-Factored



$$f(y) = \mathit{score}(\mathrm{head} = *_0, \mathrm{mod} = \mathrm{saw}_2) + \mathit{score}(\mathrm{saw}_2, \mathrm{John}_1)$$

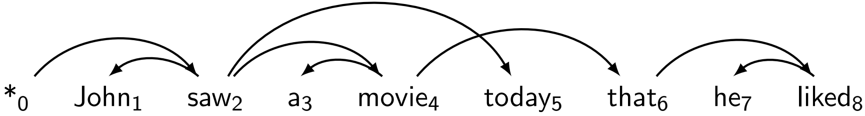
$$+\mathit{score}(\mathrm{saw}_2, \mathrm{movie}_4) + \mathit{score}(\mathrm{saw}_2, \mathrm{today}_5)$$

$$+\mathit{score}(\mathrm{movie}_4, \mathrm{a}_3) + \ldots$$

e.g. $\mathit{score}(*_0, \mathrm{saw}_2) = \log p(\mathrm{saw}_2 | *_0)$ (generative model)

or $\mathit{score}(*_0, \mathrm{saw}_2) = w \cdot \phi(\mathrm{saw}_2, *_0)$ (CRF/perceptron model)

Arc-Factored



$$f(y) = \mathit{score}(\mathrm{head} = *_0, \mathrm{mod} = \mathrm{saw}_2) + \mathit{score}(\mathrm{saw}_2, \mathrm{John}_1)$$

$$+\mathit{score}(\mathrm{saw}_2, \mathrm{movie}_4) + \mathit{score}(\mathrm{saw}_2, \mathrm{today}_5)$$

$$+\mathit{score}(\mathrm{movie}_4, \mathrm{a}_3) + \ldots$$

e.g. $\mathit{score}(*_0, \mathrm{saw}_2) = \log p(\mathrm{saw}_2 | *_0)$ (generative model)

or $\mathit{score}(*_0, \mathrm{saw}_2) = w \cdot \phi(\mathrm{saw}_2, *_0)$ (CRF/perceptron model)

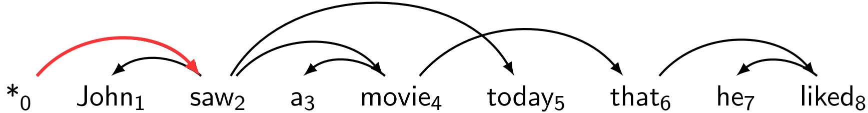
$$y^* = \arg\max_y f(y) \Leftarrow$$ Minimum Spanning Tree Algorithm

Sibling Models

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

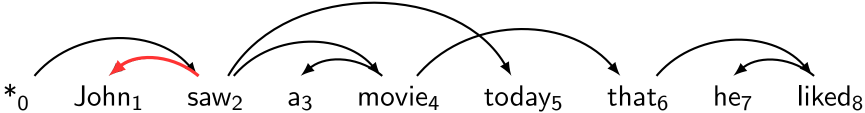
$f(y) =$

Sibling Models



$$f(y) = \mathit{score}(\mathit{head} = *_0, \mathit{prev} = \mathrm{NULL}, \mathit{mod} = \mathrm{saw}_2)$$

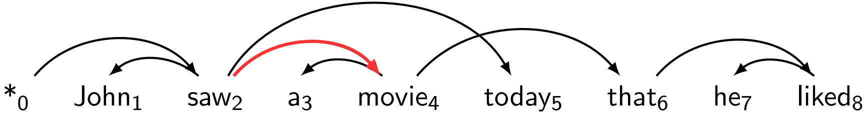
Sibling Models



$$f(y) = \textit{score}(\textit{head} = *_0, \textit{prev} = \mathrm{NULL}, \textit{mod} = \mathrm{saw}_2)$$

$$+\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1)$$

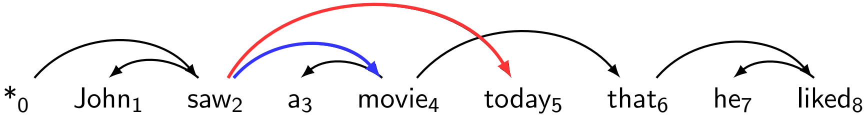
Sibling Models



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

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

Sibling Models

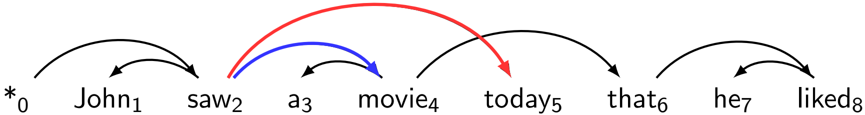


$$f(y) = \mathit{score}(\mathit{head} = *_0, \mathit{prev} = \text{NULL}, \mathit{mod} = \text{saw}_2)$$

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

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

Sibling Models



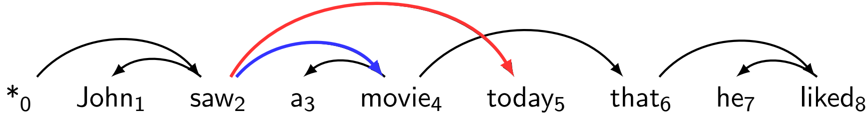
$$f(y) = \textit{score}(\textit{head} = *_0, \textit{prev} = \mathrm{NULL}, \textit{mod} = \mathrm{saw}_2)$$

$$+\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{movie}_4)$$

$$+\textit{score}(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5) + \ldots$$

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

Sibling Models



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

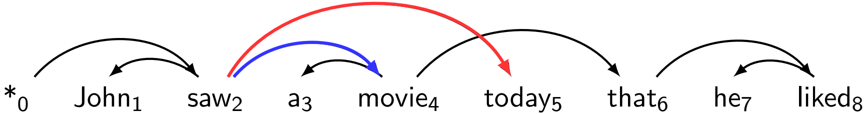
$$+\textit{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \textit{score}(\text{saw}_2, \text{NULL}, \text{movie}_4)$$

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

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

or $\textit{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) = w \cdot \phi(\text{saw}_2, \text{movie}_4, \text{today}_5)$

Sibling Models



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

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

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

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

or $\mathit{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) = w \cdot \phi(\text{saw}_2, \text{movie}_4, \text{today}_5)$

$$y^* = \arg\max_y f(y) \Leftarrow \text{NP-Hard}$$

Thought Experiment: Individual Decoding

$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

Thought Experiment: Individual Decoding

$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

Thought Experiment: Individual Decoding

$*_0$ $\quad$ John$_1$ $\quad$ saw$_2$ $\quad$ a$_3$ $\quad$ movie$_4$ $\quad$ today$_5$ $\quad$ that$_6$ $\quad$ he$_7$ $\quad$ liked$_8$

$$\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{movie}_4)$$
$$+\textit{score}(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5)$$

$$\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{that}_6)$$

Thought Experiment: Individual Decoding

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

$$\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{movie}_4)$$
$$+\textit{score}(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5)$$

$$\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{that}_6)$$

$$\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{a}_3) + \textit{score}(\mathrm{saw}_2, \mathrm{a}_3, \mathrm{he}_7)$$

Thought Experiment: Individual Decoding

$*_0$ $\quad$ John$_1$ $\quad$ saw$_2$ $\quad$ a$_3$ $\quad$ movie$_4$ $\quad$ today$_5$ $\quad$ that$_6$ $\quad$ he$_7$ $\quad$ liked$_8$

2^{n-1} **possibilities**

$$\mathit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \mathit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{movie}_4)$$
$$+\mathit{score}(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5)$$

$$\mathit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \mathit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{that}_6)$$

$$\mathit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{a}_3) + \mathit{score}(\mathrm{saw}_2, \mathrm{a}_3, \mathrm{he}_7)$$

Thought Experiment: Individual Decoding

$*_0$ $\quad$ John$_1$ $\quad$ saw$_2$ $\quad$ a$_3$ $\quad$ movie$_4$ $\quad$ today$_5$ $\quad$ that$_6$ $\quad$ he$_7$ $\quad$ liked$_8$

2^{n-1} **possibilities**

$$\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{movie}_4) + \textit{score}(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5)$$

$$\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{that}_6)$$

$$\textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{a}_3) + \textit{score}(\mathrm{saw}_2, \mathrm{a}_3, \mathrm{he}_7)$$

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

Thought Experiment Continued

$*_0$ $\quad$ John_1 $\quad$ saw_2 $\quad$ a_3 $\quad$ movie_4 $\quad$ today_5 $\quad$ that_6 $\quad$ he_7 $\quad$ liked_8

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

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

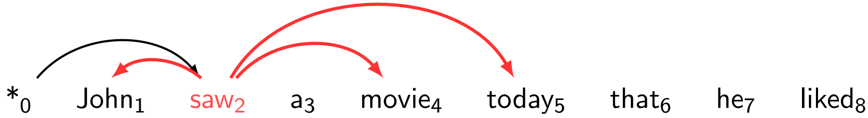
Thought Experiment Continued

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

$*_0$ $\quad$ John_1 $\quad$ saw_2 $\quad$ a_3 $\quad$ movie_4 $\quad$ today_5 $\quad$ that_6 $\quad$ he_7 $\quad$ liked_8

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

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

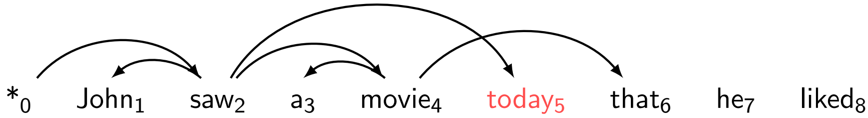
Thought Experiment Continued

$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

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

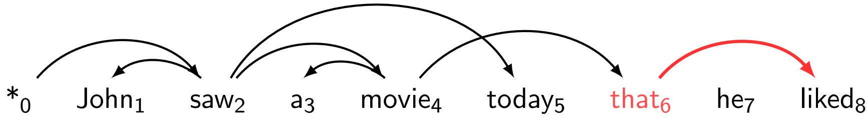
Thought Experiment Continued



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

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

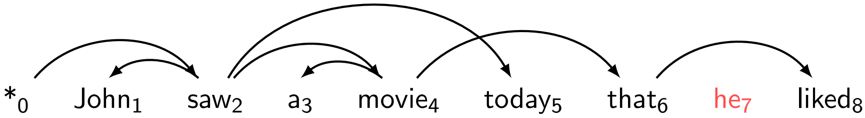
Thought Experiment Continued



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

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

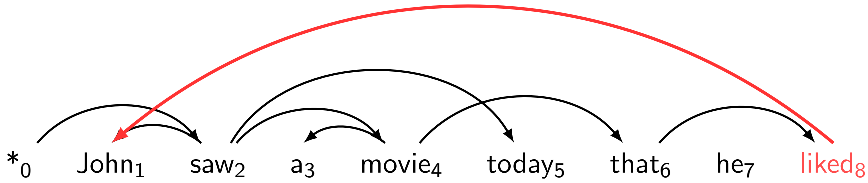
Thought Experiment Continued



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

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

Dual Decomposition Idea

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

Dual Decomposition Structure

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

Dual Decomposition Structure

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

$$\text{Rewrite as} \quad \underset{z \in \mathcal{Z},\ y \in \mathcal{Y}}{\text{argmax}} \quad f(z) + g(y)$$

$$\text{such that } z = y$$

Dual Decomposition Structure

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

$$\text{Rewrite as} \quad \operatorname*{argmax}_{\underset{\boxed{\textbf{All Possible}}}{z \in \mathcal{Z}},\ y \in \mathcal{Y}} \quad f(z) + g(y)$$

$$\text{such that } z = y$$

Dual Decomposition Structure

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

Rewrite as $\underset{z \in \mathcal{Z},\ y \in \mathcal{Y}}{\text{argmax}} \quad f(z) + g(y)$

All Possible **Valid Trees**

such that $z = y$

Dual Decomposition Structure

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

Sibling

$$\text{Rewrite as } \underset{z \in \mathcal{Z},\ y \in \mathcal{Y}}{\text{argmax}} \quad f(z) + g(y)$$

All Possible **Valid Trees**

such that $z = y$

Dual Decomposition Structure

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

$$\text{Rewrite as} \quad \underset{z \in \underbrace{\mathcal{Z}}_{\textbf{All Possible}},\ y \in \underbrace{\mathcal{Y}}_{\textbf{Valid Trees}}}{\text{argmax}} \quad \overbrace{f(z)}^{\textbf{Sibling}} + \overbrace{g(y)}^{\textbf{Arc-Factored}}$$

$$\text{such that } z = y$$

Dual Decomposition Structure

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

$$\text{Rewrite as } \quad \operatorname*{argmax}_{\underbrace{z\in\mathcal{Z}}_{\textbf{All Possible}},\ \underbrace{y\in\mathcal{Y}}_{\textbf{Valid Trees}}} \quad \overbrace{f(z)}^{\textbf{Sibling}} + \overbrace{g(y)}^{\textbf{Arc-Factored}}$$

$$\text{such that } \underbrace{z = y}_{\textbf{Constraint}}$$

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

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

Algorithm Sketch

Set penalty weights equal to 0 for all edges.

For $k = 1$ **to** K

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

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

Algorithm Sketch

Set penalty weights equal to 0 for all edges.

For $k = 1$ **to** K

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

$y^{(k)} \leftarrow$ Decode $(g(y) - \mathrm{penalty})$ 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

Penalties

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

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

Minimum Spanning Tree

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

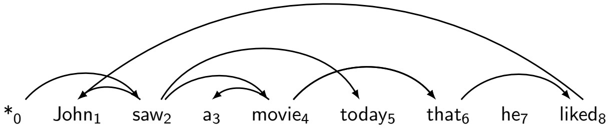
Key

$f(z)$	$\Leftarrow$	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			

Individual Decoding

Penalties

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



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

Minimum Spanning Tree

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

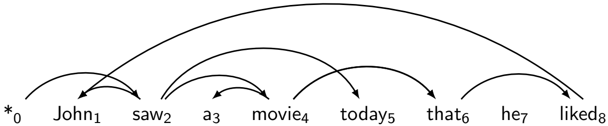
Key

$f(z)$	$\Leftarrow$	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			

Individual Decoding

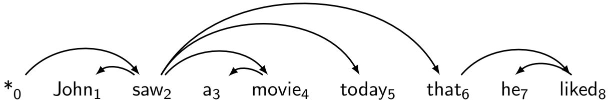
Penalties

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



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

Minimum Spanning Tree



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

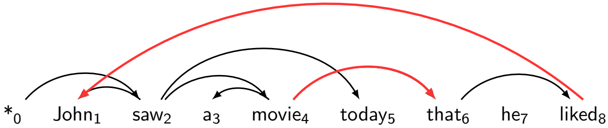
Key

$f(z)$	$\Leftarrow$	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			

Individual Decoding

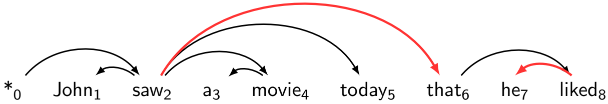
Penalties

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



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

Minimum Spanning Tree

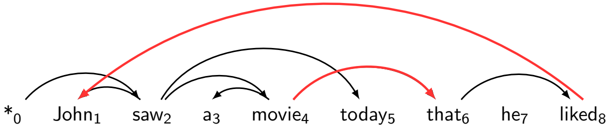


$$y^* = \arg\max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j) y(i,j))$$

Key

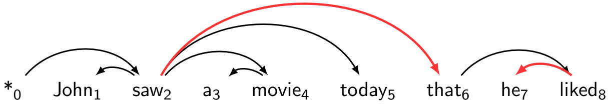
$f(z)$	$\Leftarrow$	Sibling Model	$g(y)$	$\Leftarrow$	Arc-Factored Model
$\mathcal{Z}$	$\Leftarrow$	No Constraints	$\mathcal{Y}$	$\Leftarrow$	Tree Constraints
$y(i,j) = 1$	if	y contains dependency i,j			

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

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

	Iteration 1
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

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			

Individual Decoding

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

Minimum Spanning Tree

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

Penalties

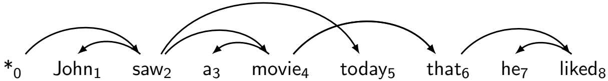
$u(i,j) = 0$ for all i,j

Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

Key

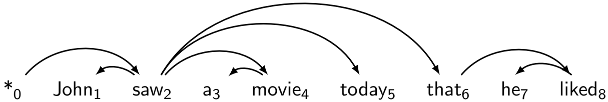
$f(z)$	$\Leftarrow$	Sibling Model	$g(y)$	$\Leftarrow$	Arc-Factored Model
$\mathcal{Z}$	$\Leftarrow$	No Constraints	$\mathcal{Y}$	$\Leftarrow$	Tree Constraints
$y(i,j) = 1$	if	y contains dependency i,j			

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

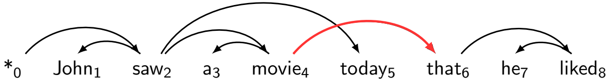
$u(i,j) = 0$ for all i,j

Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

Key

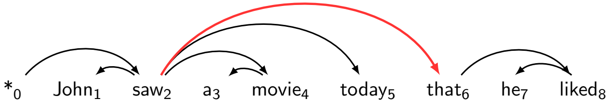
$f(z)$	$\Leftarrow$	Sibling Model	$g(y)$	$\Leftarrow$	Arc-Factored Model
$\mathcal{Z}$	$\Leftarrow$	No Constraints	$\mathcal{Y}$	$\Leftarrow$	Tree Constraints
$y(i,j) = 1$	if	y contains dependency i,j			

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

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

Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

Iteration 2	
$u(8,1)$	-1
$u(4,6)$	-2
$u(2,6)$	2
$u(8,7)$	1

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			

Individual Decoding

Penalties

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

Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

Iteration 2	
$u(8,1)$	-1
$u(4,6)$	-2
$u(2,6)$	2
$u(8,7)$	1

$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

Minimum Spanning Tree

$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

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			

Individual Decoding

$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

Minimum Spanning Tree

$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

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			

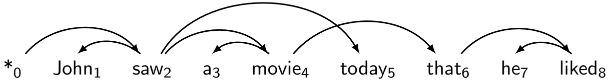
Penalties

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

Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

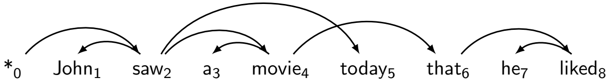
Iteration 2	
$u(8,1)$	-1
$u(4,6)$	-2
$u(2,6)$	2
$u(8,7)$	1

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

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

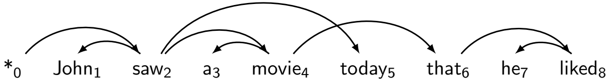
Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

Iteration 2	
$u(8,1)$	-1
$u(4,6)$	-2
$u(2,6)$	2
$u(8,7)$	1

Key

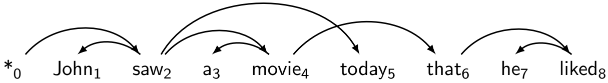
$f(z)$	$\Leftarrow$	Sibling Model	$g(y)$	$\Leftarrow$	Arc-Factored Model
$\mathcal{Z}$	$\Leftarrow$	No Constraints	$\mathcal{Y}$	$\Leftarrow$	Tree Constraints
$y(i,j) = 1$	if	y contains dependency i,j			

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

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

Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

Iteration 2	
$u(8,1)$	-1
$u(4,6)$	-2
$u(2,6)$	2
$u(8,7)$	1

Converged

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

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			

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

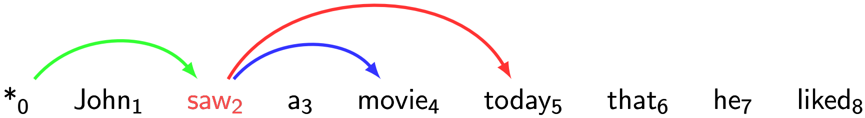
If at any iteration $y^{(k)} = z^{(k)}$, then $(y^{(k)}, z^{(k)})$ is the global optimum.

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



$$f(y) = \ldots + \mathit{score}(\mathit{gp} = *_0, \mathit{head} = \text{saw}_2, \mathit{prev} = \text{movie}_4, \mathit{mod} = \text{today}_5)$$

- Head Automata (Eisner, 2000)

  Generalization of Sibling models

  Allow arbitrary automata as local scoring function.

Roadmap

Algorithm

Experiments

Derivation

Experiments

Properties:

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

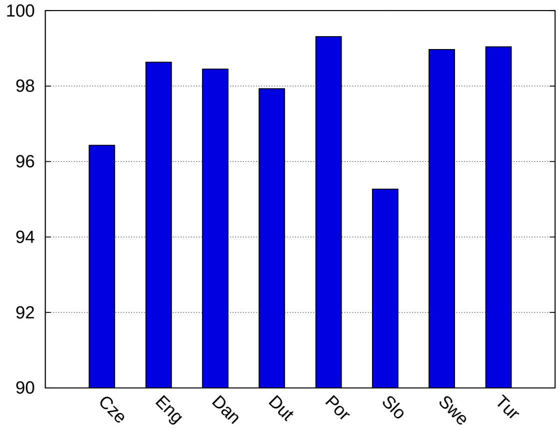
Training:

- Averaged Perceptron (more details in paper)

Experiments on:

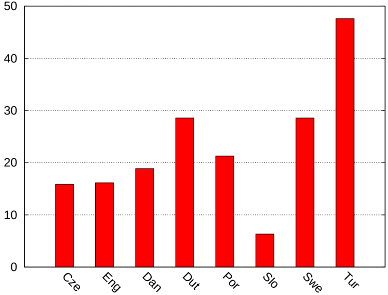
- CoNLL Datasets
- English Penn Treebank
- Czech Dependency Treebank

How often do we exactly solve the problem?

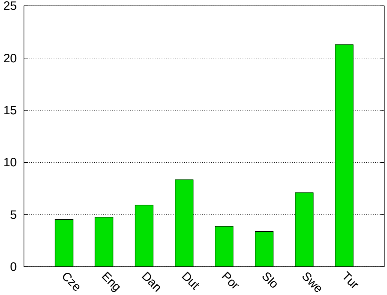


- Percentage of examples where the dual decomposition finds an exact solution.

Parsing Speed



Sibling model



Grandparent model

- Number of sentences parsed per second
- Comparable to dynamic programming for projective parsing

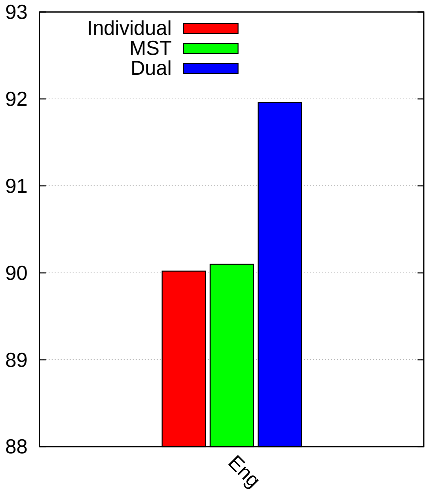
Accuracy

	Arc-Factored	Prev Best	Grandparent
Dan	89.7	91.5	**91.8**
Dut	82.3	85.6	**85.8**
Por	90.7	92.1	**93.0**
Slo	82.4	85.6	**86.2**
Swe	88.9	90.6	**91.4**
Tur	75.7	76.4	**77.6**
Eng	90.1	—	**92.5**
Cze	84.4	—	**87.3**

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 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

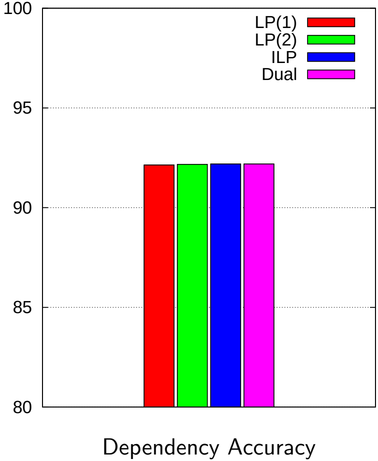
Use an LP/ILP Solver for decoding

We compare:

- Accuracy
- Exactness
- Speed

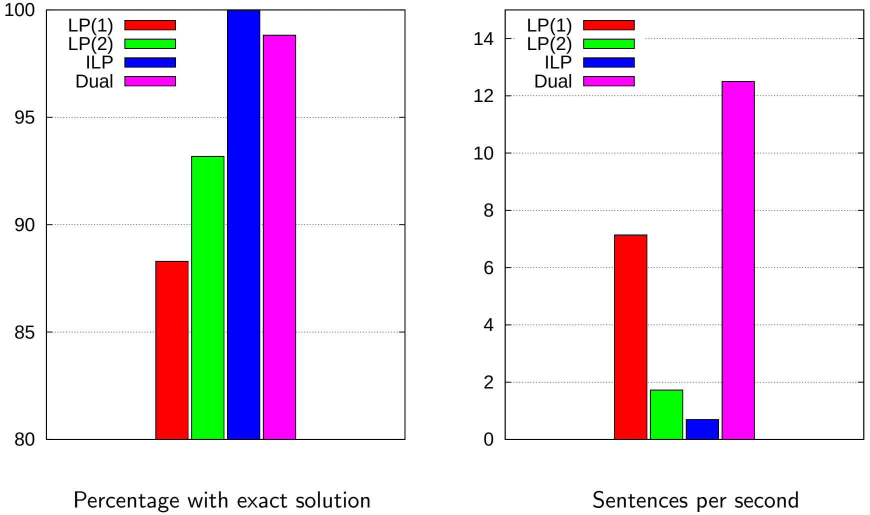
Both LP and dual decomposition methods use the same model, features, and weights w.

Comparison to LP/ILP: Accuracy



- All decoding methods have comparable accuracy

Comparison to LP/ILP: Exactness and Speed



Roadmap

Algorithm

Experiments

Derivation

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

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

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

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

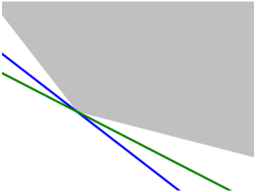
Dual is an upper bound: $L(u) \geq f(z^*) + g(y^*)$ for any u

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 $u' = u - \alpha g_u$

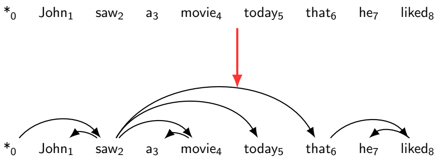
In fact, for our $L(u)$, $g_u(i,j) = z^*(i,j) - y^*(i,j)$

Related Work

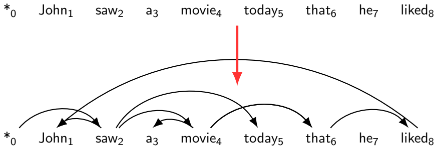
- Methods that use general purpose linear programming or integer linear programming solvers (Martins et al. 2009; Riedel and Clarke 2006; Roth and Yih 2005)
- Dual decomposition/Lagrangian relaxation in combinatorial optimization (Dantzig and Wolfe, 1960; Held and Karp, 1970; Fisher 1981)
- Dual decomposition 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

$$y^* = \arg\max_y f(y) \Leftarrow \text{NP-Hard}$$



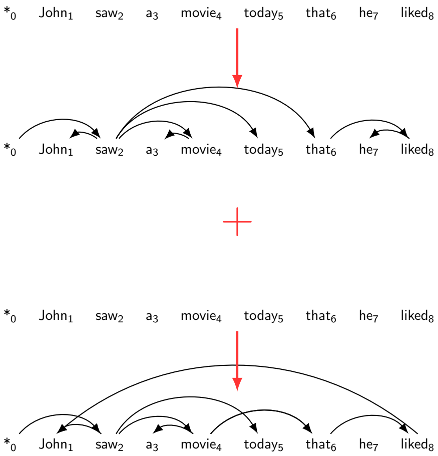
Arc-Factored Model



Sibling Model

Summary

$$y^* = \arg\max_y f(y) \Leftarrow \text{NP-Hard}$$



Arc-Factored Model

Dual Decomposition

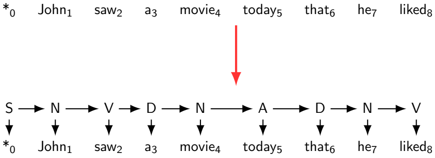
Sibling Model

Other Applications

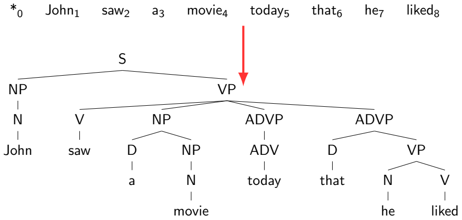
- Dual decomposition can be applied to other decoding problems.
- Rush et al. (2010) focuses on integrated dynamic programming algorithms.
  - Integrated Parsing and Tagging
  - Integrated Constituency and Dependency Parsing

Parsing and Tagging

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



HMM Model

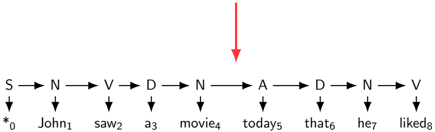


CFG Model

Parsing and Tagging

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

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

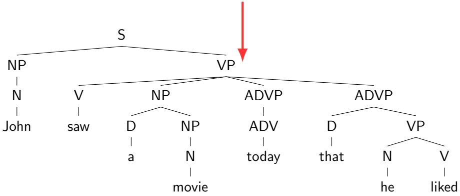


HMM Model

+

Dual Decomposition

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

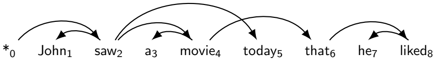


CFG Model

Dependency and Constituency

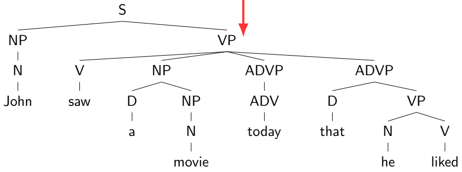
$$y^* = \arg\max_y f(y) \Leftarrow \text{Slow}$$

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$



Dependency Model

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$



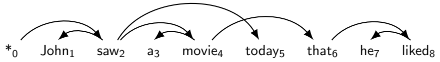
Lexicalized CFG

Dependency and Constituency

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

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

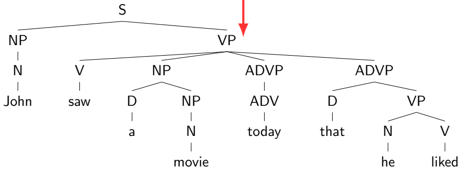
Dependency Model



Dual Decomposition

$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

Lexicalized CFG



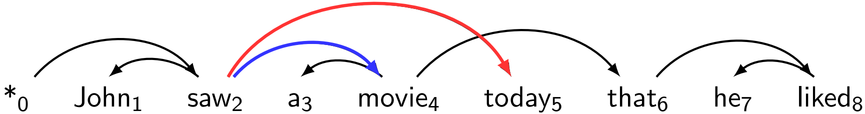
Future Directions

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

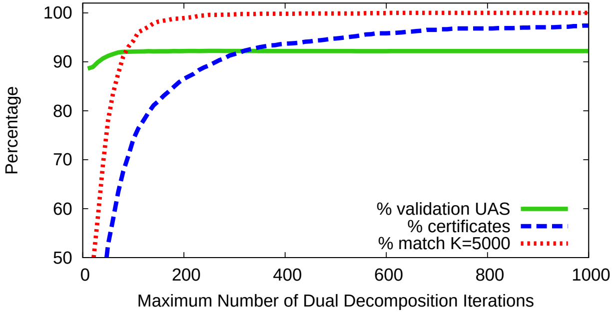
Training the Model



$$f(y) = \ldots + \mathit{score}(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5) + \ldots$$

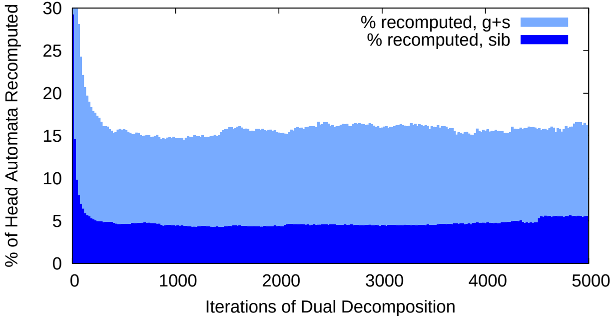
- $\mathit{score}(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5) = w \cdot \phi(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5)$
- Weight vector w trained using Averaged perceptron.
- (More details in the paper.)

Early Stopping



Early Stopping

Caching



Caching speed