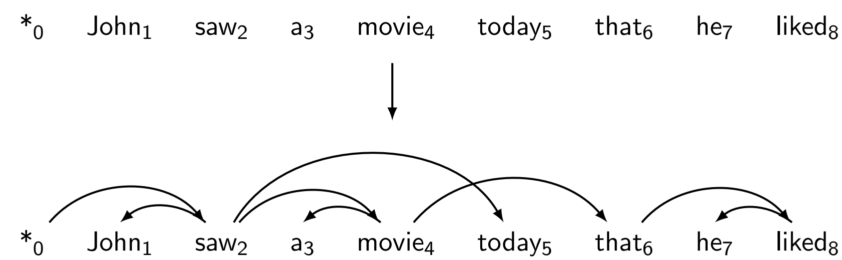
Dual Decomposition for Parsing with Non-Projective Head Automata

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

Non-Projective Dependency Parsing



Important problem in many languages.

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

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

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

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

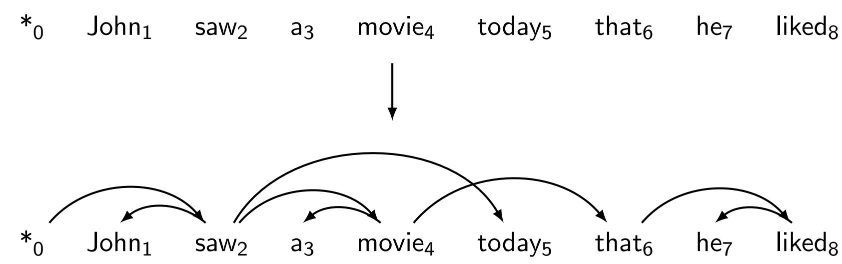
Efficient - Faster than previously proposed algorithms

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

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

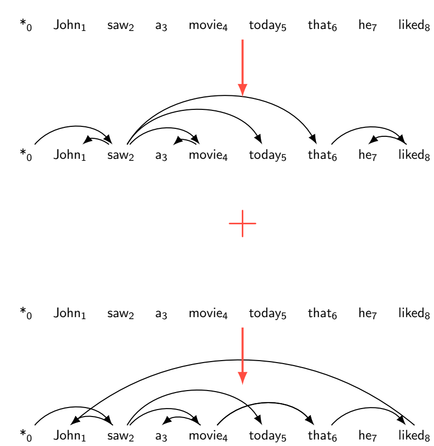
Roadmap

Algorithm

Experiments

Derivation

Algorithm Outline

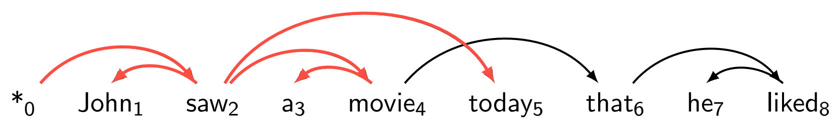


Arc-Factored Model

Dual Decomposition

Sibling Model

Arc-Factored



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

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

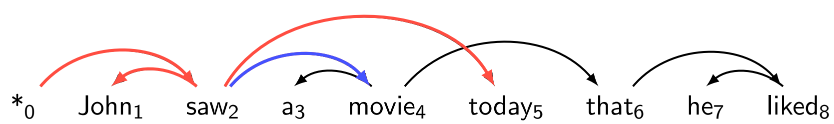
$+\textit{score}(\text{movie}_4, \text{a}_3) + \ldots$

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

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

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

Sibling Models



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

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

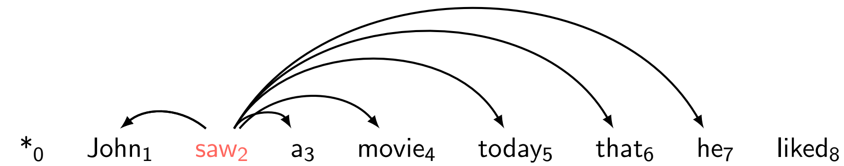
$+\textit{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) + \ldots$

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

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

Thought Experiment: Individual Decoding

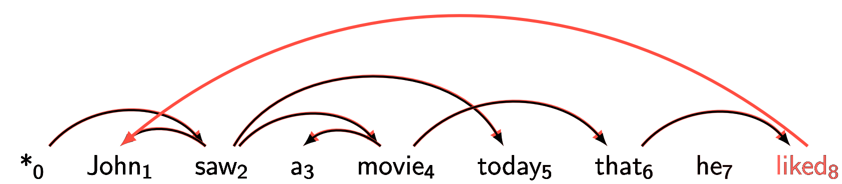


2^{n-1} **possibilities**:

- $\textit{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \textit{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) + \textit{score}(\text{saw}_2, \text{movie}_4, \text{today}_5)$
- $\textit{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \textit{score}(\text{saw}_2, \text{NULL}, \text{that}_6)$
- $\textit{score}(\text{saw}_2, \text{NULL}, \text{a}_3) + \textit{score}(\text{saw}_2, \text{a}_3, \text{he}_7)$

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.

But we might violate some constraints.

Dual Decomposition Idea

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

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 $y^{(k)}(i,j) - z^{(k)}(i,j)$

Dual Decomposition Structure

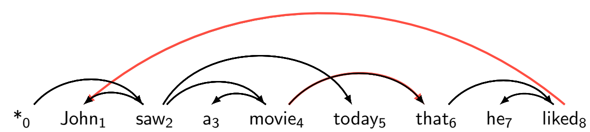
Goal $y^* = \arg\max_{y \in \mathcal{Y}} f(y)$

Rewrite as $\operatorname{argmax}_{z \in \mathcal{Z},\, y \in \mathcal{Y}} \; f(z) + g(y)$

such that $z = y$

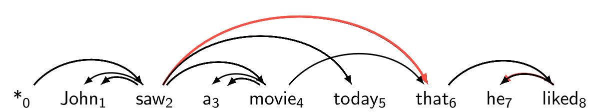
Sibling → $f(z)$; **Arc-Factored** → $g(y)$; **All Possible** → $z \in \mathcal{Z}$; **Valid Trees** → $y \in \mathcal{Y}$; **Constraint** → $z = y$

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.

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

Roadmap

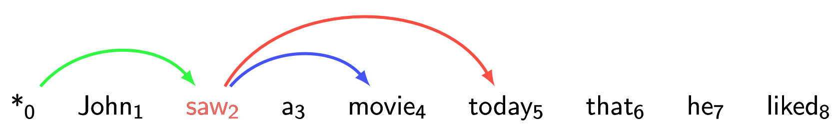
Algorithm

Experiments

Derivation

Extensions

- Grandparent Models



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

- 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

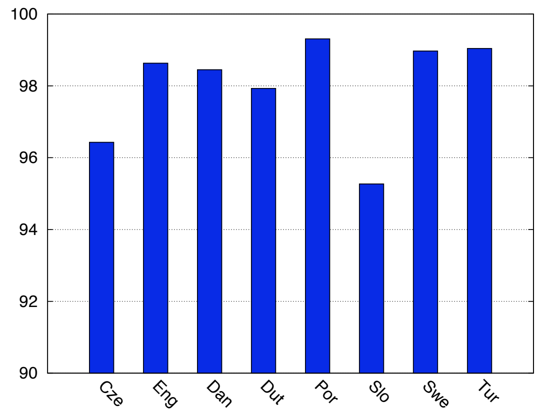
Training:

- 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.

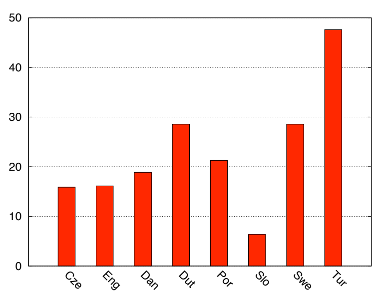
Accuracy

	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**

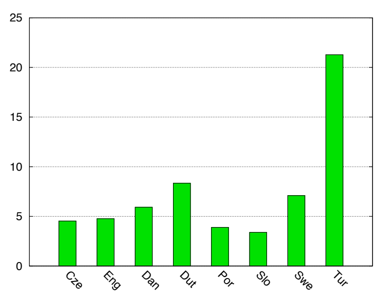
Prev Best - Best reported results for CoNLL-X data set, includes

- Approximate search (McDonald and Pereira, 2006)
- Loop belief propagation (Smith and Eisner, 2008)
- (Integer) Linear Programming (Martins et al., 2009)

Parsing Speed



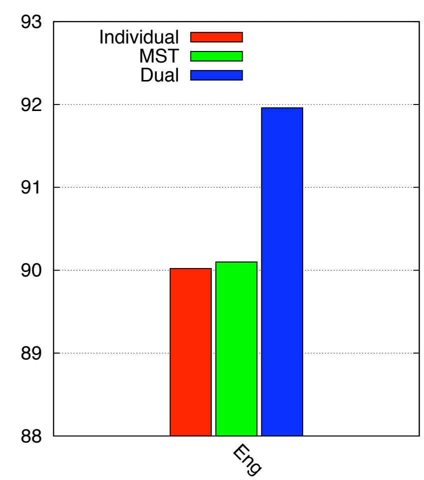
Sibling model



Grandparent model

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

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

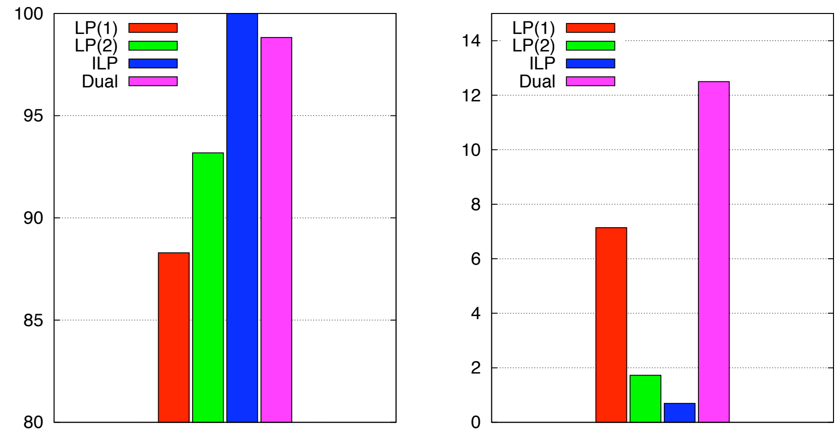
Use an LP/ILP Solver for decoding

We compare:

- Accuracy
- Exactness
- Speed

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

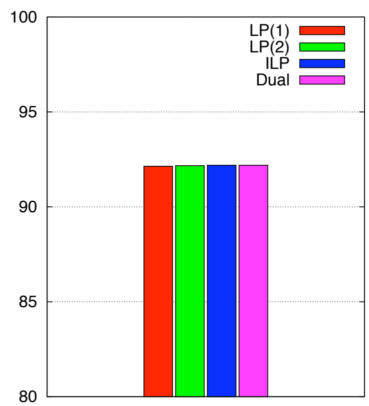
Comparison to LP/ILP: Exactness and Speed



Percentage with exact solution

Sentences per second

Comparison to LP/ILP: Accuracy



Dependency Accuracy

- All decoding methods have comparable accuracy

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

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

Related Work

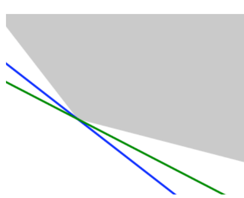
- Methods that use general purpose linear programming or integer linear programming solvers (Martins et al. 2009; Riedel and Clarke 2006; Roth and Yih 2005)
- Dual decomposition/Lagrangian relaxation in combinatorial optimization (Dantzig and Wolfe, 1960; Held and Karp, 1970; Fisher 1981)
- Dual decomposition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

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,

$$L(v) \geq L(u) + g_u \cdot (v - u)$$

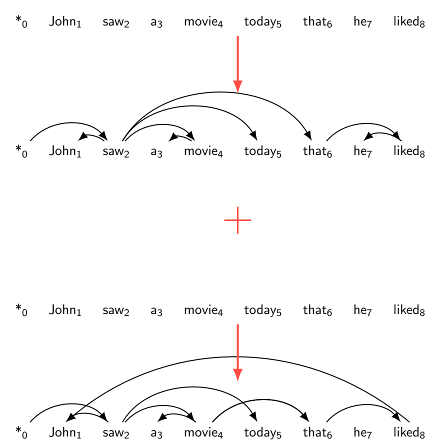


Subgradient methods use updates $u' = u - \alpha g_u$

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

Summary

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



Arc-Factored Model

Dual Decomposition

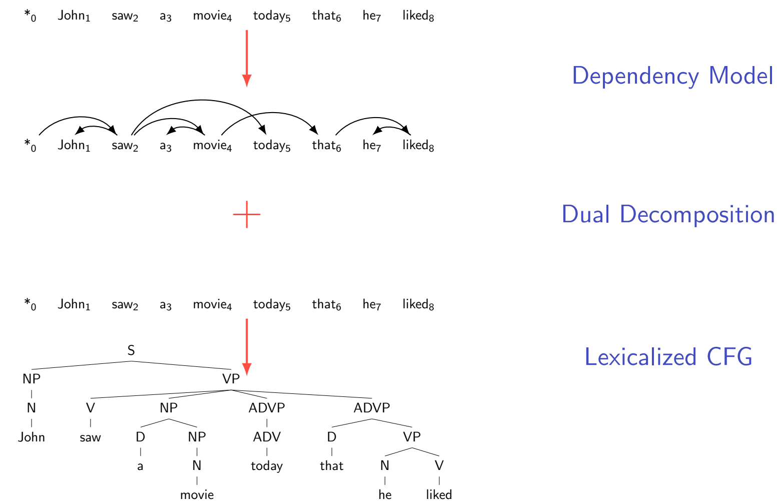
Sibling Model

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

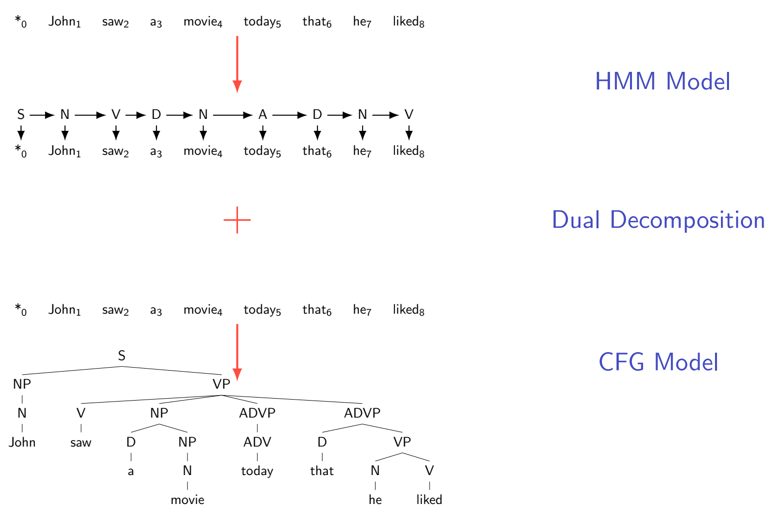
Dependency and Constituency

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



Parsing and Tagging

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



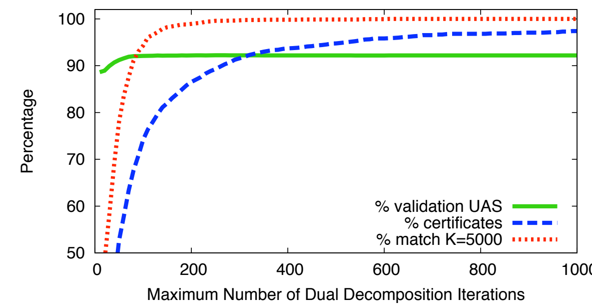
Future Directions

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?

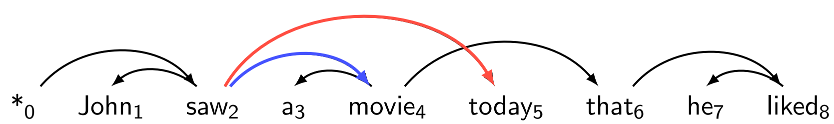
Appendix

Early Stopping



Early Stopping

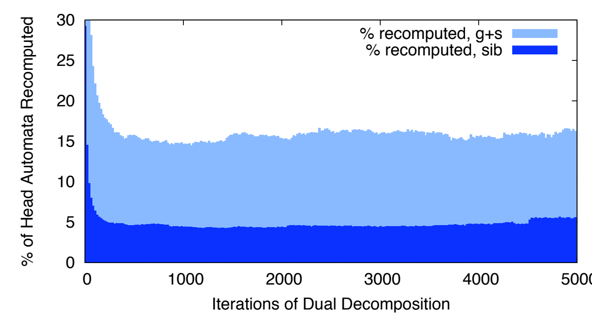
Training the Model



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

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

Caching



Caching speed