Dual Decomposition for Natural Language Processing

Alexander M. Rush and Michael Collins

Decoding complexity

focus: decoding problem for natural language tasks

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

motivation:

- richer model structure often leads to improved accuracy
- exact decoding for complex models tends to be intractable

Decoding tasks

many common problems are intractable to decode exactly

high complexity

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

NP-Hard

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

in practice:

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

Motivation

cannot hope to find exact algorithms (particularly when NP-Hard)

aim: develop decoding algorithms with formal guarantees

method:

- derive fast algorithms that provide certificates of optimality
- show that for practical instances, these algorithms often yield exact solutions
- provide strategies for improving solutions or finding approximate solutions when no certificate is found

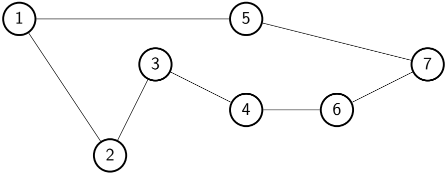
dual decomposition helps us develop algorithms of this form

Lagrangian relaxation (Held and Karp, 1971)

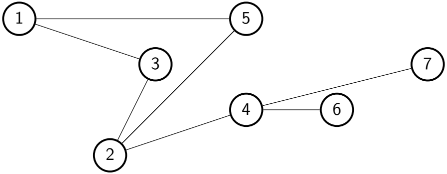
important method from combinatorial optimization

initially used for traveling salesman problems

optimal tour - NP-Hard



optimal 1-tree - easy (MST)



Dual decomposition (Komodakis et al., 2010; Lemaréchal, 2001)

goal: solve complicated optimization problem

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

method: decompose into subproblems, solve iteratively

benefit: can choose decomposition to provide "easy" subproblems

aim for simple and efficient combinatorial algorithms

- dynamic programming
- minimum spanning tree
- shortest path
- min-cut
- bipartite match
- etc.

Related work

there are related methods used NLP with similar motivation
related methods:

- belief propagation (particularly max-product) (Smith and Eisner, 2008)
- factored A* search (Klein and Manning, 2003)
- exact coarse-to-fine (Raphael, 2001)

aim to find exact solutions without exploring the full search space

Tutorial outline

focus:

- developing dual decomposition algorithms for new NLP tasks
- understanding formal guarantees of the algorithms
- extensions to improve exactness and select solutions

outline:

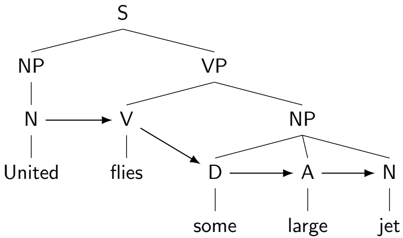
1. worked algorithm for combined parsing and tagging
2. important theorems and formal derivation
3. more examples from parsing, sequence labeling, MT
4. practical considerations for implementing dual decomposition
5. relationship to linear programming relaxations
6. further variations and advanced examples

1. Worked example

aim: walk through a dual decomposition algorithm for combined parsing and part-of-speech tagging

- introduce formal notation for parsing and tagging
- give assumptions necessary for decoding
- step through a run of the dual decomposition algorithm

Combined parsing and part-of-speech tagging



goal: find parse tree that optimizes

$$\textit{score}(\mathrm{S} \ \rightarrow \mathrm{NP\ VP}) + \textit{score}(\mathrm{VP} \ \rightarrow \mathrm{V\ NP}) +$$

$$\ldots + \textit{score}(\mathrm{N} \rightarrow \mathrm{V}) + \textit{score}(\mathrm{N} \rightarrow \mathrm{United}) + \ldots$$

Constituency parsing

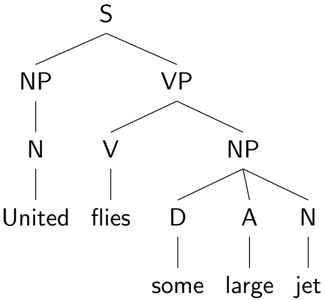
notation:

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

goal:

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

example: a context-free grammar for constituency parsing



Part-of-speech tagging

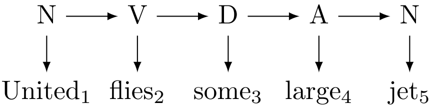
notation:

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

goal:

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

example: an HMM for part-of speech tagging

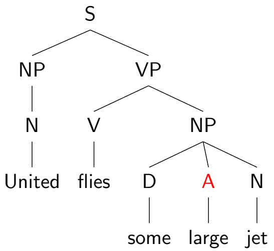


Identifying tags

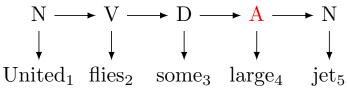
notation: identify the tag labels selected by each model

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

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



y



z

Combined optimization

goal:

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

such that for all $i = 1 \ldots n$, $t \in \mathcal{T}$,

$$y(i, t) = z(i, t)$$

i.e. find the best parse and tagging pair that agree on tag labels

equivalent formulation:

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

where $l : \mathcal{Y} \rightarrow \mathcal{Z}$ extracts the tag sequence from a parse tree

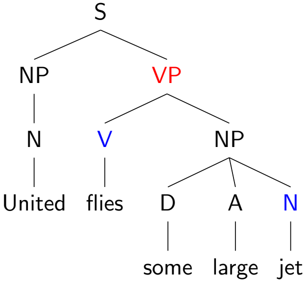
Dynamic programming intersection

can solve by solving the product of the two models

example:

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

replace $\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP}$
with
$\mathrm{S}_{\mathrm{N,N}} \rightarrow \mathrm{NP}_{\mathrm{N,N}}\mathrm{VP}_{\mathrm{V,N}}$



Parsing assumption

the structure of $\mathcal{Y}$ could be CFG, TAG, etc.

assumption: optimization with u can be solved efficiently

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

generally benign since u can be incorporated into the structure of f

example: CFG with rule scoring function h

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

where

$$\begin{aligned}
\arg\max_{y \in \mathcal{Y}} \quad & f(y) + \sum_{i,t} u(i,t) y(i,t) = \\
\arg\max_{y \in \mathcal{Y}} \quad & \sum_{X \to Y\ Z \in y} h(X \to Y\ Z) + \sum_{(i,X) \in y} (h(X \to w_i) + u(i,X))
\end{aligned}$$

Tagging assumption

we make a similar assumption for the set $\mathcal{Z}$

assumption: optimization with u can be solved efficiently

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

example: HMM with scores for transitions T and observations O

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

where

$$\begin{aligned} \arg\max_{z \in \mathcal{Z}} \quad & g(z) - \sum_{i,t} u(i,t) z(i,t) = \\ \arg\max_{z \in \mathcal{Z}} \quad & \sum_{t \to t' \in z} T(t \to t') + \sum_{(i,t) \in z} (O(t \to w_i) - u(i,t)) \end{aligned}$$

Dual decomposition algorithm

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

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

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

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

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

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

CKY Parsing

Penalties

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

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

Viterbi Decoding

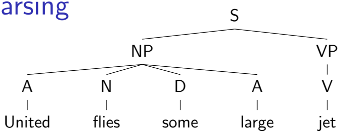
United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Key

$f(y)$	$\Leftarrow$	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



Penalties

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

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

Viterbi Decoding

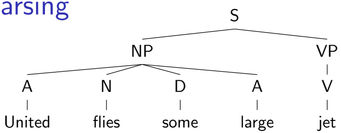
United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Key

$f(y)$	$\Leftarrow$	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

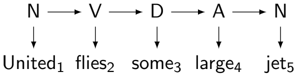

S
NP VP
A N D A V
United flies some large jet

Penalties

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

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

Viterbi Decoding

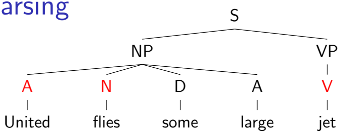

N V D A N
United_1 flies_2 some_3 large_4 jet_5

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

Key

$f(y)$	$\Leftarrow$	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

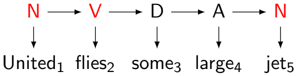


$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,t} u(i,t) y(i,t))$$

Penalties

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

Viterbi Decoding

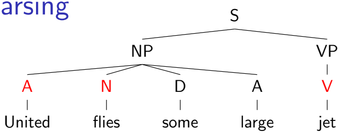


$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,t} u(i,t) z(i,t))$$

Key

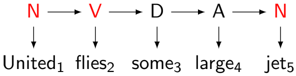
$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



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

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

$u(i,t) = 0$ 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

CKY Parsing

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

Viterbi Decoding

United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Penalties

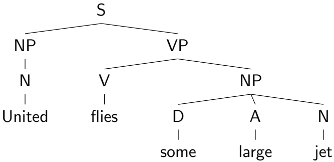
$u(i,t) = 0$ for all i,t

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

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



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

Viterbi Decoding

United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Key

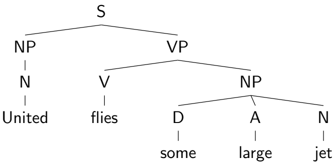
$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

Penalties

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

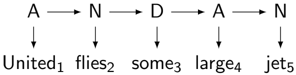
Iteration 1	
$u(1,A)$	-1
$u(1,N)$	1
$u(2,N)$	-1
$u(2,V)$	1
$u(5,V)$	-1
$u(5,N)$	1

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

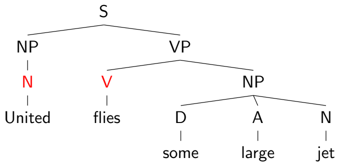
$u(i,t) = 0$ for all i,t

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

Key

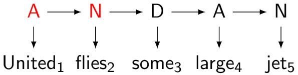
$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	HMM
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Taggings
$y(i,t) = 1$	if	y contains tag t at position i			

CKY Parsing



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

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

$u(i,t) = 0$ 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

CKY Parsing

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

Viterbi Decoding

United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Penalties

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

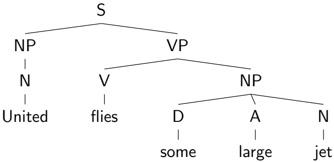
Iteration 1	
$u(1, A)$	-1
$u(1, N)$	1
$u(2, N)$	-1
$u(2, V)$	1
$u(5, V)$	-1
$u(5, N)$	1

Iteration 2	
$u(5, V)$	-1
$u(5, N)$	1

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



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

Viterbi Decoding

United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Key

$f(y)$	$\Leftarrow$	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			

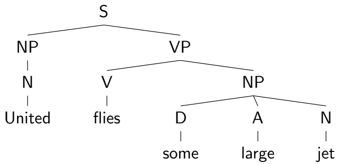
Penalties

$u(i,t) = 0$ 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

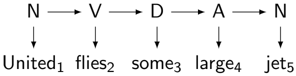
Iteration 2	
$u(5, V)$	-1
$u(5, N)$	1

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			

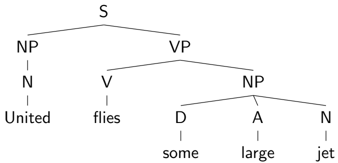
Penalties

$u(i,t) = 0$ 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

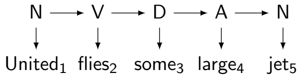
Iteration 2	
$u(5,V)$	-1
$u(5,N)$	1

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

$u(i,t) = 0$ 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

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

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			

Main theorem

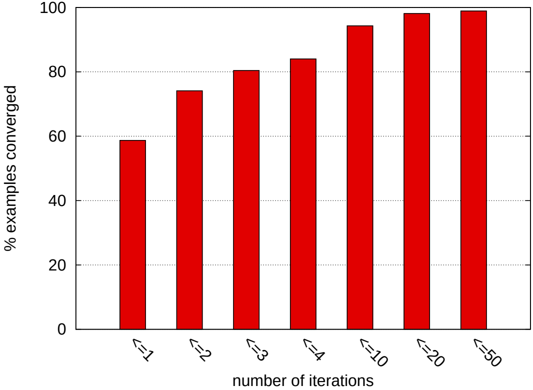
theorem: if at any iteration, for all i, $t \in \mathcal{T}$

$$y^{(k)}(i,t) = z^{(k)}(i,t)$$

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

proof: focus of the next section

Convergence



2. Formal properties

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

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

Lagrangian

goal:

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

Lagrangian:

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

redistribute terms

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

Lagrangian dual

Lagrangian:

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

Lagrangian dual:

$$\begin{aligned} L(u) &= \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} L(u, y, z) \\ &= \max_{y \in \mathcal{Y}} \left(f(y) + \sum_{i,t} u(i,t)y(i,t)\right) + \\ &\quad\ \max_{z \in \mathcal{Z}} \left(g(z) - \sum_{i,t} u(i,t)z(i,t)\right) \end{aligned}$$

Theorem 1. Upper bound

define:

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

theorem: for any value of u

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

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

note: upper bound may be useful as input to branch and bound or A* search

Theorem 1. Upper bound (proof)

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

proof:

$$
\begin{aligned}
L(u) &= \max_{y \in \mathcal{Y}, z \in \mathcal{Z}} L(u, y, z) && (1) \\
&\geq \max_{y \in \mathcal{Y}, z \in \mathcal{Z} : y = z} L(u, y, z) && (2) \\
&= \max_{y \in \mathcal{Y}, z \in \mathcal{Z} : y = z} f(y) + g(z) && (3) \\
&= f(y^*) + g(z^*) && (4)
\end{aligned}
$$

Formal algorithm (reminder)

```
Set $u^{(1)}(i,t) = 0$ for all $i,\ t \in \mathcal{T}$
**For** $k = 1$ **to** $K$
    $y^{(k)} \leftarrow \arg\max_{y \in \mathcal{Y}} f(y) + \sum_{i,t} u^{(k)}(i,t) y(i,t)$ [Parsing]
    $z^{(k)} \leftarrow \arg\max_{z \in \mathcal{Z}} g(z) - \sum_{i,t} u^{(k)}(i,t) z(i,t)$ [Tagging]
    **If** $y^{(k)}(i,t) = z^{(k)}(i,t)$ for all $i, t$ **Return** $(y^{(k)}, z^{(k)})$
    **Else** $u^{(k+1)}(i,t) \leftarrow u^{(k)}(i,t) - \alpha_k (y^{(k)}(i,t) - z^{(k)}(i,t))$
```

Theorem 2. Convergence

notation:

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

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

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

we have

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

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

proof: by subgradient convergence (next section)

Dual solutions

define:

- for any value of u

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

and

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

- y_u and z_u are the dual solutions for a given u

Theorem 3. Optimality

theorem: if there exists u such that

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

for all i, t then

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

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

$$(y_u, z_u)$$

Theorem 3. Optimality (proof)

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

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

proof: by the definitions of y_u and z_u

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

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

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

but y^* and z^* are optimal

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

Dual optimization

Lagrangian dual:

$$\begin{aligned} L(u) &= \max_{y\in\mathcal{Y},z\in\mathcal{Z}} L(u,y,z) \\ &= \max_{y\in\mathcal{Y}} \left(f(y) + \sum_{i,t} u(i,t)y(i,t) \right) + \\ & \quad \max_{z\in\mathcal{Z}} \left(g(z) - \sum_{i,t} u(i,t)z(i,t) \right) \end{aligned}$$

goal: dual problem is to find the tightest upper bound

$$\min_u L(u)$$

Dual subgradient

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

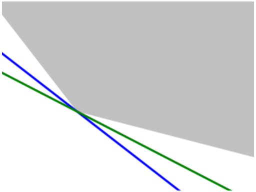
properties:

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

handle non-differentiability by using subgradient descent

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

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



Subgradient algorithm

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

recall, y_u and z_u are the argmax's of the two terms

subgradient:

$$g_u(i,t) \;=\; y_u(i,t)-z_u(i,t)$$

subgradient descent: move along the subgradient

$$u'(i,t)=u(i,t)-\alpha\left(y_u(i,t)-z_u(i,t)\right)$$

guaranteed to find a minimum with conditions given earlier for α

3. More examples

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

- context-free parsing combined with dependency parsing
- corpus-level part-of-speech tagging
- combined translation alignment

Combined constituency and dependency parsing

(Rush et al., 2010)

setup: assume separate models trained for constituency and dependency parsing

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

example:

- combine lexicalized CFG with second-order dependency parser

Lexicalized constituency parsing

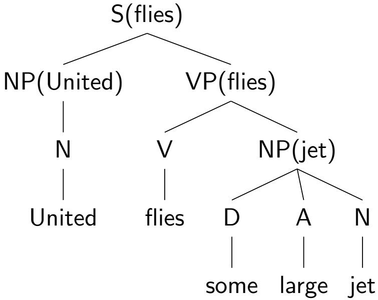
notation:

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

goal:

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

example: a lexicalized context-free grammar

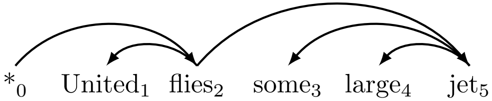


Dependency parsing

define:

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

example:



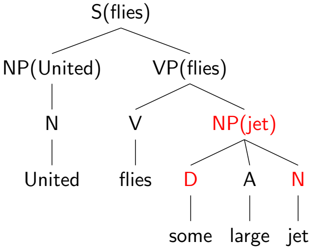
$*_0$ United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

Identifying dependencies

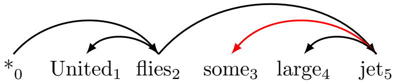
notation: identify the dependencies selected by each model

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

example: a constituency and dependency parse with $y(3, 5) = 1$ and $z(3, 5) = 1$



y



z

Combined optimization

goal:

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

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

$$y(i,j) = z(i,j)$$

CKY Parsing

Penalties

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

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

Dependency Parsing

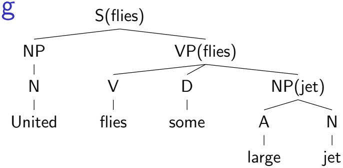
$*_0$ United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing



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

Penalties

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

Dependency Parsing

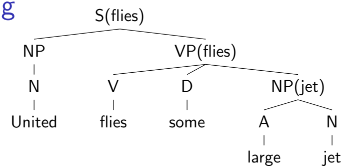
$*_0$ United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing

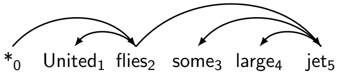


$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Penalties

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

Dependency Parsing

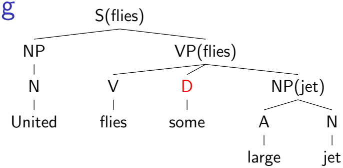


$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing

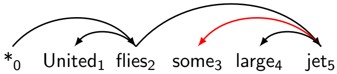


$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Penalties

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

Dependency Parsing

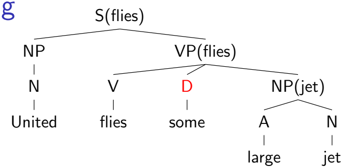


$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing



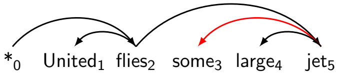
$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Penalties

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

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

Dependency Parsing



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

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing

Penalties

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

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

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

Dependency Parsing

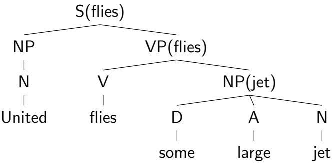
$*_0$ United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing



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

Penalties

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

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

Dependency Parsing

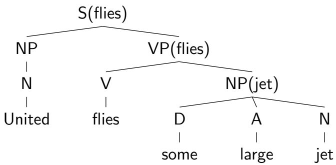
$*_0$ United$_1$ flies$_2$ some$_3$ large$_4$ jet$_5$

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

Key

$f(y)$	⇐	CFG	$g(z)$	⇐	Dependency Model
$\mathcal{Y}$	⇐	Parse Trees	$\mathcal{Z}$	⇐	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing



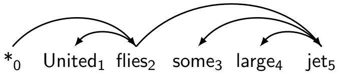
$$y^* = \arg\max_{y \in \mathcal{Y}} (f(y) + \sum_{i,j} u(i,j) y(i,j))$$

Penalties

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

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

Dependency Parsing

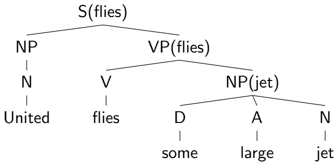


$$z^* = \arg\max_{z \in \mathcal{Z}} (g(z) - \sum_{i,j} u(i,j) z(i,j))$$

Key

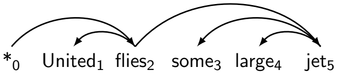
$f(y)$	⇐	CFG	$g(z)$	⇐	Dependency Model
$\mathcal{Y}$	⇐	Parse Trees	$\mathcal{Z}$	⇐	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

CKY Parsing



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

Dependency Parsing



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

Penalties

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

Iteration 1	
$u(2,3)$	-1
$u(5,3)$	1

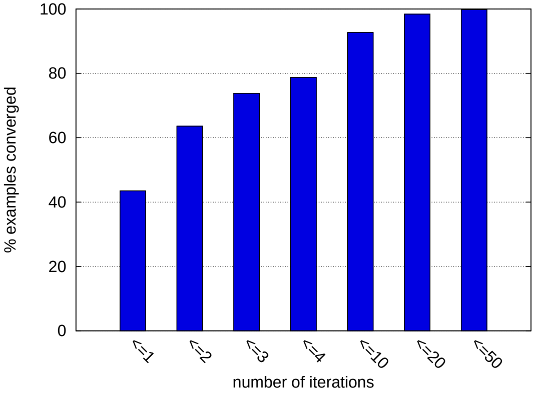
Converged

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

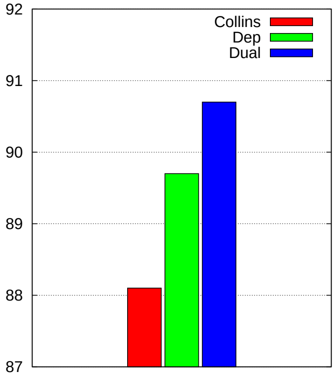
Key

$f(y)$	$\Leftarrow$	CFG	$g(z)$	$\Leftarrow$	Dependency Model
$\mathcal{Y}$	$\Leftarrow$	Parse Trees	$\mathcal{Z}$	$\Leftarrow$	Dependency Trees
$y(i,j) = 1$	if	y contains dependency i,j			

Convergence



Integrated Constituency and Dependency Parsing: Accuracy



F$_1$ Score

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

Corpus-level tagging

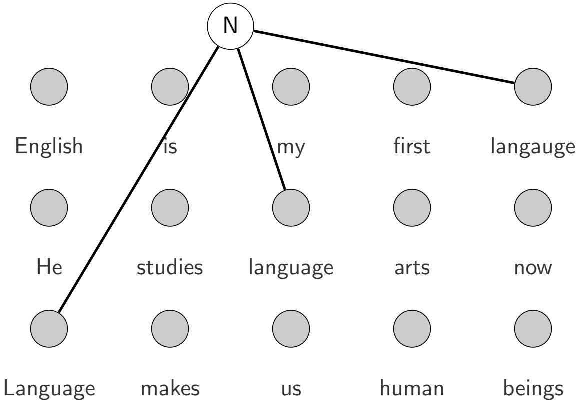
setup: given a corpus of sentences and a trained sentence-level tagging model

problem: find best tagging for each sentence, while at the same time enforcing inter-sentence soft constraints

example:

- test-time decoding with a trigram tagger
- constraint that each word type prefer a single POS tag

Corpus-level tagging



Sentence-level decoding

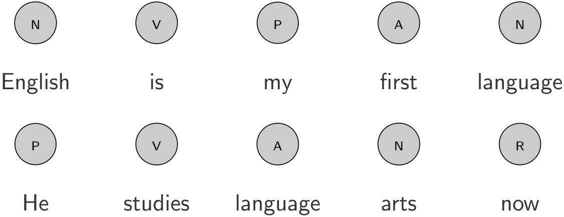
notation:

- $\mathcal{Y}_i$ is set of tag sequences for input sentence i
- $\mathcal{Y} = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_m$ is set of tag sequences for the input corpus
- $Y \in \mathcal{Y}$ is a valid tag sequence for the corpus
- $F(Y) = \sum_i f(Y_i)$ is the score for tagging the whole corpus

goal:

$$\arg\max_{Y \in \mathcal{Y}} F(Y)$$

example: decode each sentence with a trigram tagger



N V P A N

English is my first language

P V A N R

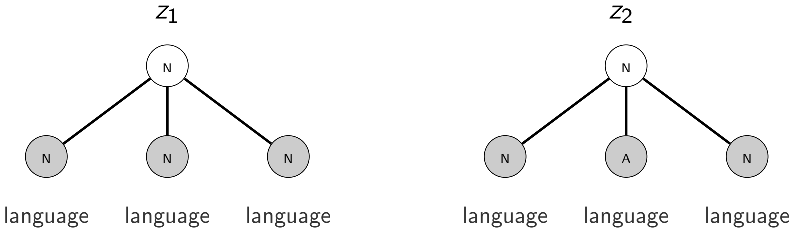
He studies language arts now

Inter-sentence constraints

notation:

- $\mathcal{Z}$ is set of possible assignments of tags to word types
- $z \in \mathcal{Z}$ is a valid tag assignment
- $g(z)$ is a scoring function for assignments to word types

example: an MRF model that encourages words of the same type to choose the same tag



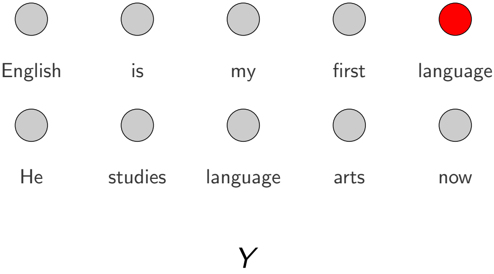
$$g(z_1) > g(z_2)$$

Identifying word tags

notation: identify the tag labels selected by each model

- $Y_s(i,t) = 1$ when the tagger for sentence s at position i selects tag t
- $z(s,i,t) = 1$ when the constraint assigns at sentence s position i the tag t

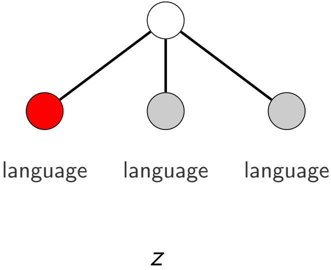
example: a parse and tagging with $Y_1(5,N) = 1$ and $z(1,5,N) = 1$



English is my first language

He studies language arts now

Y



language language language

z

Combined optimization

goal:

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

such that for all $s = 1 \ldots m$, $i = 1 \ldots n$, $t \in \mathcal{T}$,

$$Y_s(i, t) = z(s, i, t)$$

Penalties

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

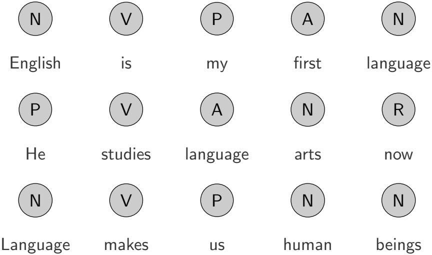
Tagging

MRF

Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Tagging



N V P A N

English is my first language

P V A N R

He studies language arts now

N V P N N

Language makes us human beings

MRF

Penalties

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

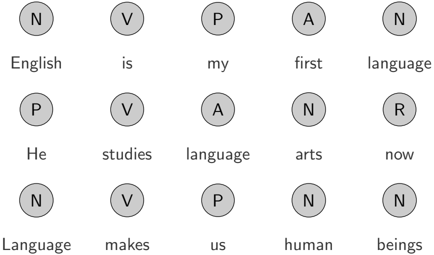
Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Penalties

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

Tagging


N V P A N

English is my first language

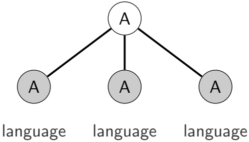
P V A N R

He studies language arts now

N V P N N

Language makes us human beings

MRF


A

A A A

language language language

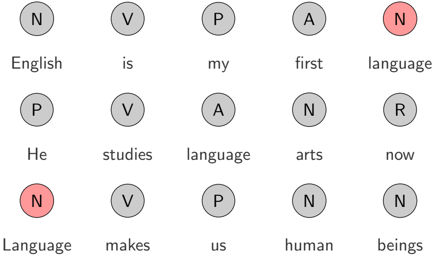
Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

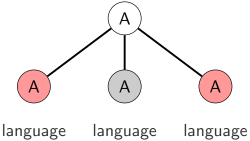
Penalties

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

Tagging


N V P A N
English is my first language
P V A N R
He studies language arts now
N V P N N
Language makes us human beings

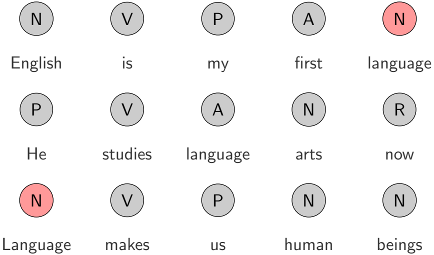
MRF



Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Tagging

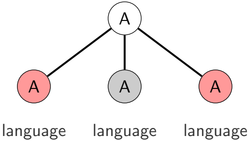


Penalties

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

Iteration 1	
$u(1, 5, N)$	-1
$u(1, 5, A)$	1
$u(3, 1, N)$	-1
$u(3, 1, A)$	1

MRF



Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Tagging

Penalties

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

Iteration 1	
$u(1, 5, N)$	-1
$u(1, 5, A)$	1
$u(3, 1, N)$	-1
$u(3, 1, A)$	1

MRF

Key

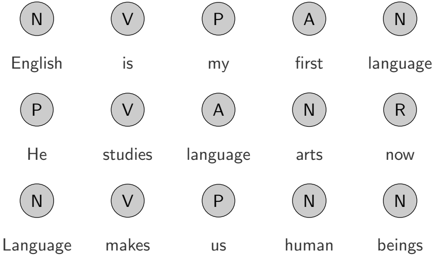
$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Penalties

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

Iteration 1	
$u(1, 5, N)$	-1
$u(1, 5, A)$	1
$u(3, 1, N)$	-1
$u(3, 1, A)$	1

Tagging



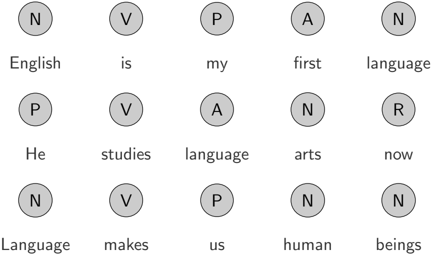
N	V	P	A	N
English	is	my	first	language
P	V	A	N	R
He	studies	language	arts	now
N	V	P	N	N
Language	makes	us	human	beings

MRF

Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Tagging



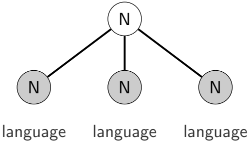
N	V	P	A	N
English	is	my	first	language
P	V	A	N	R
He	studies	language	arts	now
N	V	P	N	N
Language	makes	us	human	beings

Penalties

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

Iteration 1	
$u(1, 5, N)$	-1
$u(1, 5, A)$	1
$u(3, 1, N)$	-1
$u(3, 1, A)$	1

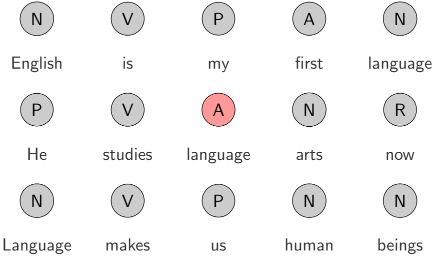
MRF



Key

$F(Y)$	⇐	Tagging model	$g(z)$	⇐	MRF
$\mathcal{Y}$	⇐	Sentence-level tagging	$\mathcal{Z}$	⇐	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Tagging



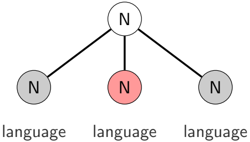
N V P A N
English is my first language
P V A N R
He studies language arts now
N V P N N
Language makes us human beings

Penalties

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

Iteration 1	
$u(1, 5, N)$	-1
$u(1, 5, A)$	1
$u(3, 1, N)$	-1
$u(3, 1, A)$	1

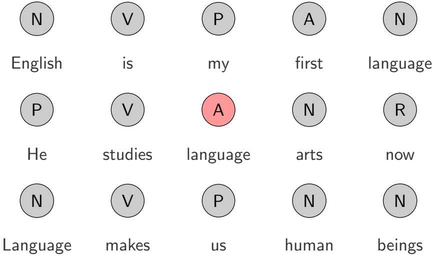
MRF



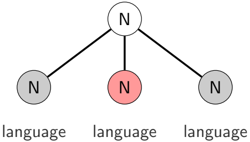
Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Tagging


N V P A N
English is my first language
P V A N R
He studies language arts now
N V P N N
Language makes us human beings

MRF



Penalties

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

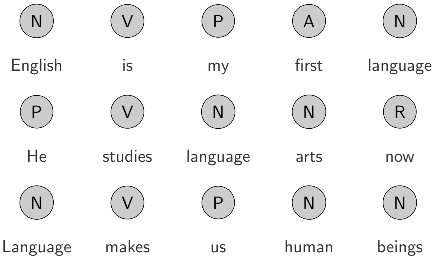
Iteration 1	
$u(1, 5, N)$	-1
$u(1, 5, A)$	1
$u(3, 1, N)$	-1
$u(3, 1, A)$	1

Iteration 2	
$u(1, 5, N)$	-1
$u(1, 5, A)$	1
$u(3, 1, N)$	-1
$u(3, 1, A)$	1
$u(2, 3, N)$	1
$u(2, 3, A)$	-1

Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Tagging



N	V	P	A	N
English	is	my	first	language
P	V	N	N	R
He	studies	language	arts	now
N	V	P	N	N
Language	makes	us	human	beings

MRF

Penalties

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

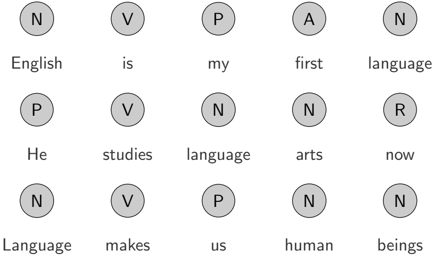
Iteration 1	
$u(1,5,N)$	-1
$u(1,5,A)$	1
$u(3,1,N)$	-1
$u(3,1,A)$	1

Iteration 2	
$u(1,5,N)$	-1
$u(1,5,A)$	1
$u(3,1,N)$	-1
$u(3,1,A)$	1
$u(2,3,N)$	1
$u(2,3,A)$	-1

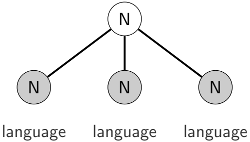
Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i,t) = 1$	if	sentence s has tag t at position i			

Tagging


N V P A N
English is my first language
P V N N R
He studies language arts now
N V P N N
Language makes us human beings

MRF



Penalties

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

Iteration 1	
$u(1, 5, N)$	-1
$u(1, 5, A)$	1
$u(3, 1, N)$	-1
$u(3, 1, A)$	1

Iteration 2	
$u(1, 5, N)$	-1
$u(1, 5, A)$	1
$u(3, 1, N)$	-1
$u(3, 1, A)$	1
$u(2, 3, N)$	1
$u(2, 3, A)$	-1

Key

$F(Y)$	$\Leftarrow$	Tagging model	$g(z)$	$\Leftarrow$	MRF
$\mathcal{Y}$	$\Leftarrow$	Sentence-level tagging	$\mathcal{Z}$	$\Leftarrow$	Inter-sentence constraints
$Y_s(i, t) = 1$	if	sentence s has tag t at position i			

Combined alignment (DeNero and Macherey, 2011)

setup: assume separate models trained for English-to-French and French-to-English alignment

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

example:

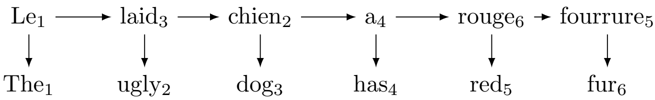
- HMM models for both directional alignments (assume correct alignment is one-to-one for simplicity)

English-to-French alignment

define:

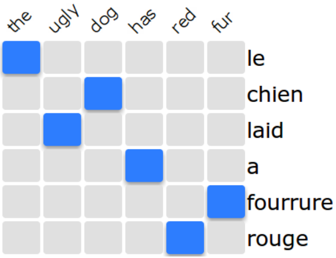
- $\mathcal{Y}$ is set of all possible English-to-French alignments
- $y \in \mathcal{Y}$ is a valid alignment
- $f(y)$ scores of the alignment

example: HMM alignment



$\text{Le}_1 \rightarrow \text{laid}_3 \rightarrow \text{chien}_2 \rightarrow \text{a}_4 \rightarrow \text{rouge}_6 \rightarrow \text{fourrure}_5$

The_1 ugly_2 dog_3 has_4 red_5 fur_6



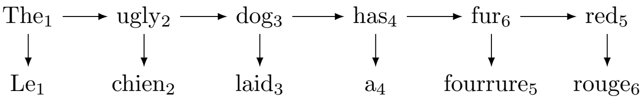
the	ugly	dog	has	red	fur	
■						le
		■				chien
	■					laid
			■			a
					■	fourrure
				■		rouge

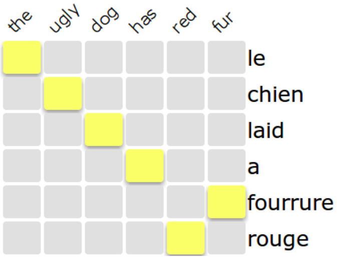
French-to-English alignment

define:

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

example: HMM alignment





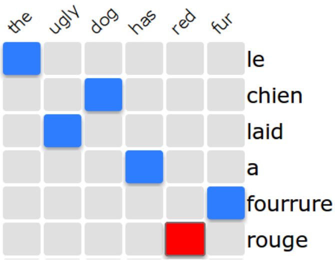
the	ugly	dog	has	red	fur	
🟨						le
	🟨					chien
		🟨				laid
			🟨			a
					🟨	fourrure
				🟨		rouge

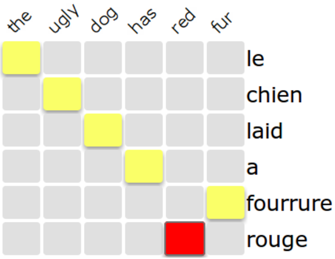
Identifying word alignments

notation: identify the tag labels selected by each model

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

example: two HMM alignment models with $y(6,5) = 1$ and $z(6,5) = 1$





Combined optimization

goal:

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

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

$$y(i,j) = z(i,j)$$

English-to-French

Penalties

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

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

French-to-English

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

Key

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

English-to-French

the	ugly	dog	has	red	fur	
■						le
		■				chien
	■					laid
			■			a
					■	fourrure
				■		rouge

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

Penalties

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

French-to-English

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

Key

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

English-to-French

the	ugly	dog	has	red	fur	
🟦						le
		🟦				chien
	🟦					laid
			🟦			a
					🟦	fourrure
				🟦		rouge

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

French-to-English

the	ugly	dog	has	red	fur	
🟨						le
	🟨					chien
		🟨				laid
			🟨			a
					🟨	fourrure
				🟨		rouge

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

Penalties

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

Key

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

English-to-French

	the	ugly	dog	has	red	fur
le	🟦					
chien			🟥			
laid		🟥				
a				🟦		
fourrure						🟦
rouge					🟦	

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

French-to-English

	the	ugly	dog	has	red	fur
le	🟨					
chien		🟥				
laid			🟥			
a				🟨		
fourrure						🟨
rouge					🟨	

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

Penalties

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

Key

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

English-to-French

the	ugly	dog	has	red	fur	
🟦						le
		🟥				chien
	🟥					laid
			🟦			a
					🟦	fourrure
				🟦		rouge

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

French-to-English

the	ugly	dog	has	red	fur	
🟨						le
	🟥					chien
		🟥				laid
			🟨			a
					🟨	fourrure
				🟨		rouge

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

Penalties

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

Iteration 1	
$u(3,2)$	-1
$u(2,2)$	1
$u(2,3)$	-1
$u(3,3)$	1

Key

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

English-to-French

Penalties

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

Iteration 1	
$u(3,2)$	-1
$u(2,2)$	1
$u(2,3)$	-1
$u(3,3)$	1

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

French-to-English

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

Key

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

English-to-French

	the	ugly	dog	has	red	fur
le	🟦					
chien			🟦			
laid		🟦				
a				🟦		
fourrure						🟦
rouge					🟦	

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

French-to-English

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

Penalties

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

	Iteration 1
$u(3,2)$	-1
$u(2,2)$	1
$u(2,3)$	-1
$u(3,3)$	1

Key

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

English-to-French

	the	ugly	dog	has	red	fur
le	🟦					
chien			🟦			
laid		🟦				
a				🟦		
fourrure						🟦
rouge					🟦	

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

French-to-English

	the	ugly	dog	has	red	fur
le	🟨					
chien			🟨			
laid		🟨				
a				🟨		
fourrure						🟨
rouge					🟨	

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

Penalties

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

	Iteration 1
$u(3,2)$	-1
$u(2,2)$	1
$u(2,3)$	-1
$u(3,3)$	1

Key

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

English-to-French

	the	ugly	dog	has	red	fur
le	🟦					
chien			🟦			
laid		🟦				
a				🟦		
fourrure						🟦
rouge					🟦	

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

French-to-English

	the	ugly	dog	has	red	fur
le	🟨					
chien			🟨			
laid		🟨				
a				🟨		
fourrure						🟨
rouge					🟨	

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

Penalties

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

	Iteration 1
$u(3,2)$	-1
$u(2,2)$	1
$u(2,3)$	-1
$u(3,3)$	1

Key

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

English-to-French

	the	ugly	dog	has	red	fur
le	🟦					
chien			🟦			
laid		🟦				
a				🟦		
fourrure						🟦
rouge					🟦	

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

French-to-English

	the	ugly	dog	has	red	fur
le	🟨					
chien			🟨			
laid		🟨				
a				🟨		
fourrure						🟨
rouge					🟨	

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

Penalties

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

	Iteration 1
$u(3,2)$	-1
$u(2,2)$	1
$u(2,3)$	-1
$u(3,3)$	1

Key

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

4. Practical issues

aim: overview of practical dual decomposition techniques

- tracking the progress of the algorithm
- choice of update rate α_k
- lazy update of dual solutions
- extracting solutions if algorithm does not converge

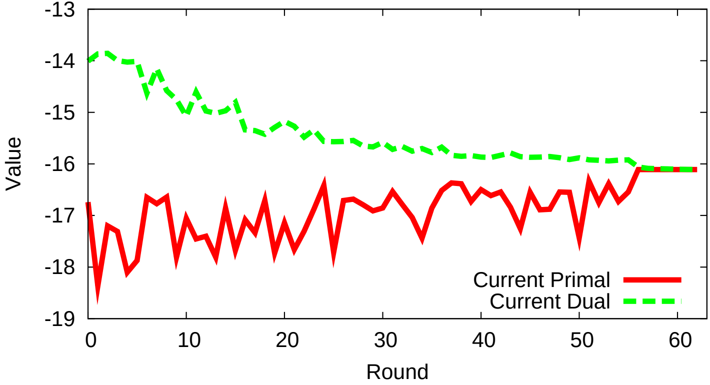
Optimization tracking

at each stage of the algorithm there are several useful values

track:

- $y^{(k)}$, $z^{(k)}$ are current dual solutions
- $L(u^{(k)})$ is the current dual value
- $y^{(k)}$, $l(y^{(k)})$ is a potential primal feasible solution
- $f(y^{(k)}) + g(l(y^{(k)}))$ is the potential primal value

Tracking example



example run from syntactic machine translation (later in talk)

- current primal

$$f(y^{(k)}) + g(l(y^{(k)}))$$

- current dual

$$L(u^{(k)})$$

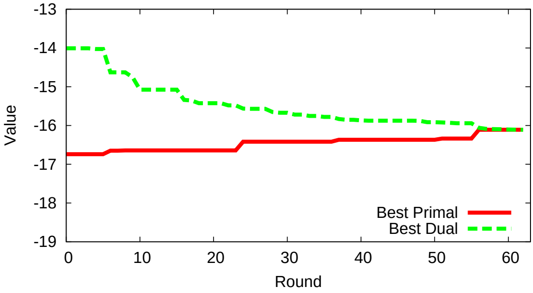
Optimization progress

useful signals:

- $L(u^{(k)}) - L(u^{(k-1)})$ is the dual change (may be positive)
- $\min_k L(u^{(k)})$ is the best dual value (tightest upper bound)
- $\max_k f(y^{(k)}) + g(l(y^{(k)}))$ is the best primal value

the optimal value must be between the best dual and primal values

Progress example

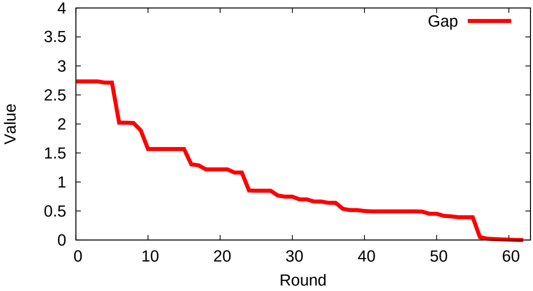


best primal

$$\max_k f(y^{(k)}) + g(l(y^{(k)}))$$

best dual

$$\min_k L(u^{(k)})$$



gap

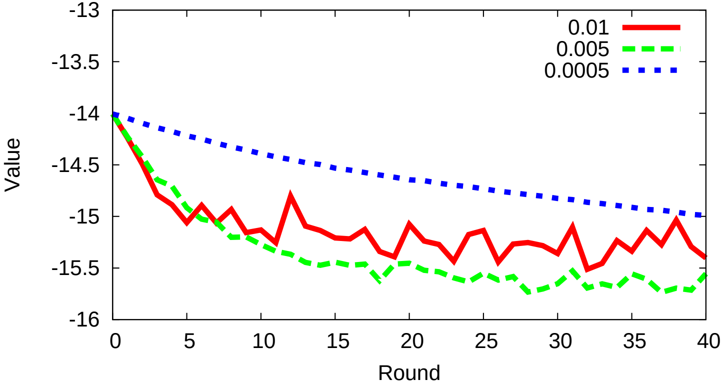
$$\min_k L(u^k) \quad -$$

$$\max_k f(y^{(k)}) \quad + \quad g(l(y^{(k)})$$

Update rate

choice of α_k has important practical consequences

- α_k too high causes dual value to fluctuate
- α_k too low means slow progress



Update rate

practical: find a rate that is robust to varying inputs

- $\alpha_k = c$ (constant rate) can be very fast, but hard to find constant that works for all problems
- $\alpha_k = \frac{c}{k}$ (decreasing rate) often cuts rate too aggressively, lowers value even when making progress
- rate based on dual progress
  - $\alpha_k = \frac{c}{t+1}$ where $t < k$ is number of iterations where dual value increased
  - robust in practice, reduces rate when dual value is fluctuating

Lazy decoding

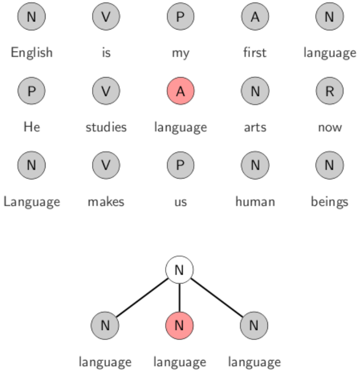
idea: don't recompute $y^{(k)}$ or $z^{(k)}$ from scratch each iteration

lazy decoding: if subgradient $u^{(k)}$ is sparse, then $y^{(k)}$ may be very easy to compute from $y^{(k-1)}$

use:

- helpful if y or z factor naturally into independent components
- can be important for fast decompositions

Lazy decoding example


N V P A N
English is my first language
P V A N R
He studies language arts now
N V P N N
Language makes us human beings
N
N N N
language language language

recall corpus-level tagging example

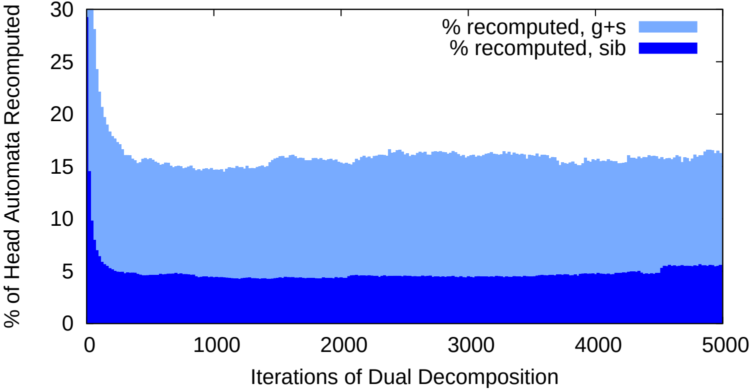
at this iteration, only sentence 2 receives a weight update

with lazy decoding

$$Y_1^{(k)} \leftarrow Y_1^{(k-1)}$$
$$Y_3^{(k)} \leftarrow Y_3^{(k-1)}$$

Lazy decoding results

lazy decoding is critical for the efficiency of some applications



recomputation statistics for non-projective dependency parsing

Approximate solution

upon agreement the solution is exact, but this may not occur

otherwise, there is an easy way to find an approximate solution

choose: the structure $y^{(k')}$ where

$$k' = \arg\max_k f(y^{(k)}) + g(l(y^{(k)}))$$

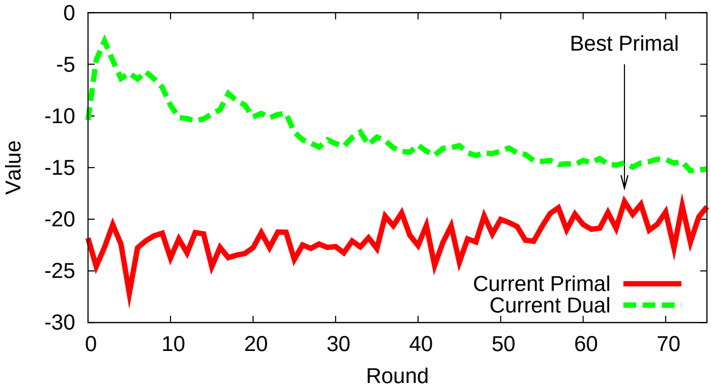
is the iteration with the best primal score

guarantee: the solution $y^{k'}$ is non-optimal by at most

$$(\min_k L(u^k)) - (f(y^{(k')}) + g(l(y^{(k')})))$$

there are other methods to estimate solutions, for instance by averaging solutions (see Nedić and Ozdaglar (2009))

Choosing best solution

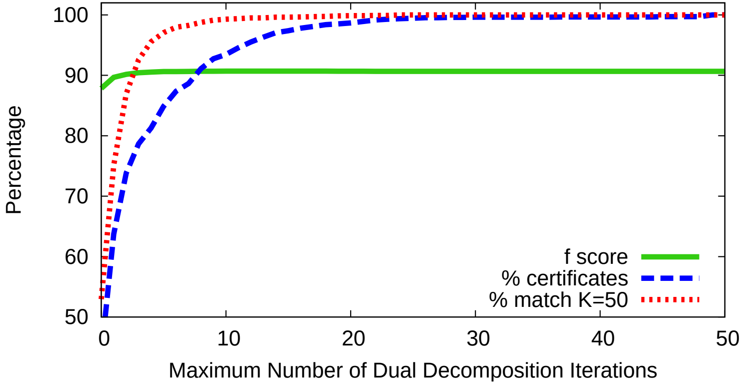


non-exact example from syntactic translation

best approximate primal solution occurs at iteration 63

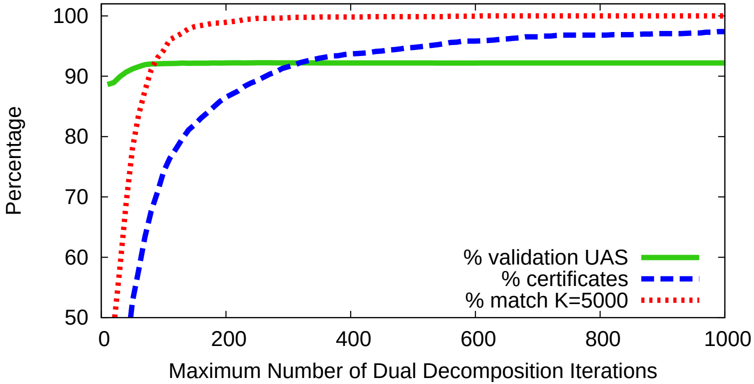
Early stopping results

early stopping results for constituency and dependency parsing



Early stopping results

early stopping results for non-projective dependency parsing



Tightening

instead of using approximate solution, can tighten the algorithm

may help find an exact solution at the cost of added complexity

this technique is the focus of the next section

5. Linear programming

aim: explore the connections between dual decomposition and linear programming

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

Simplex

define:

- $\Delta_y \subset \mathcal{R}^{|\mathcal{Y}|}$ is the simplex over $\mathcal{Y}$ where $\alpha \in \Delta_y$ implies

$$\alpha_y \geq 0 \text{ and } \sum_y \alpha_y = 1$$

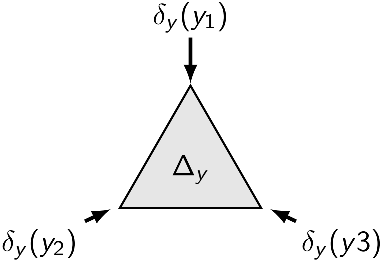
- α is distribution over $\mathcal{Y}$
- Δ_z is the simplex over $\mathcal{Z}$
- $\delta_y : \mathcal{Y} \to \Delta_y$ maps elements to the simplex

example:

$$\mathcal{Y} = \{y_1, y_2, y_3\}$$

vertices

- $\delta_y(y_1) = (1, 0, 0)$
- $\delta_y(y_2) = (0, 1, 0)$
- $\delta_y(y_3) = (0, 0, 1)$



Theorem 1. Simplex linear program

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

goal: optimize linear program

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

theorem:

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

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

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

linear program has optimum at extreme point

note: finding the highest scoring distribution α over $\mathcal{Y}$

proof shows that best distribution chooses a single parse

Combined linear program

optimize over the simplices Δ_y and Δ_z instead of the discrete sets $\mathcal{Y}$ and $\mathcal{Z}$

goal: optimize linear program

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

such that for all i, t

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

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

the best distributions α^*,β^* are possibly no longer a single parse tree or tag sequence

Lagrangian

Lagrangian:

$$
\begin{aligned}
M(u,\alpha,\beta) &= \sum_y \alpha_y f(y) + \sum_z \beta_z g(z) + \sum_{i,t} u(i,t) \left(\sum_y \alpha_y y(i,t) - \sum_z \beta_z z(i,t) \right) \\
&= \left(\sum_y \alpha_y f(y) + \sum_{i,t} u(i,t) \sum_y \alpha_y y(i,t) \right) + \\
& \quad \left(\sum_z \beta_z g(z) - \sum_{i,t} u(i,t) \sum_z \beta_z z(i,t) \right)
\end{aligned}
$$

Lagrangian dual:

$$M(u) = \max_{\alpha \in \Delta_y, \beta \in \Delta_z} M(u,\alpha,\beta)$$

Theorem 2. Strong duality

define:

- α^*, β^* is the optimal assignment to α, β in the linear program

theorem:

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

proof: by linear programming duality

Theorem 3. Dual relationship

theorem: for any value of u,

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

note: solving the original Lagrangian dual also solves dual of the linear program

Theorem 3. Dual relationship (proof sketch)

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

$$\begin{aligned} L(u) &= \max_{y \in \mathcal{Y}} \left(f(y) + \sum_{i,t} u(i,t) y(i,t) \right) + \ldots \\ M(u) &= \max_{\alpha \in \Delta_y} \left(\sum_{y} \alpha_y f(y) + \sum_{i,t} u(i,t) \sum_{y} \alpha_y y(i,t) \right) + \ldots \end{aligned}$$

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

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

Summary

$f(y) + g(z)$	original primal objective
$L(u)$	original dual
$\sum_y \alpha_y f(y) + \sum_z \beta_z g(z)$	LP primal objective
$M(u)$	LP dual

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

$$\min_u M(u) = \min_u L(u) = \sum_y \alpha_y^* f(y) + \sum_z \beta_z^* g(z)$$

Primal relationship

define:

- $\mathcal{Q} \subseteq \Delta_y \times \Delta_z$ corresponds to feasible solutions of the original problem

$$\begin{aligned}\mathcal{Q} = \{(\delta_y(y), \delta_z(z))\colon & y \in \mathcal{Y}, z \in \mathcal{Z}, \\ & y(i,t) = z(i,t) \text{ for all } (i,t)\}\end{aligned}$$

- $\mathcal{Q}' \subseteq \Delta_y \times \Delta_z$ is the set of feasible solutions to the LP

$$\begin{aligned}\mathcal{Q}' = \{(\alpha, \beta)\colon & \alpha \in \Delta_{\mathcal{Y}}, \beta \in \Delta_{\mathcal{Z}}, \\ & \textstyle\sum_y \alpha_y y(i,t) = \sum_z \beta_z z(i,t) \text{ for all } (i,t)\}\end{aligned}$$

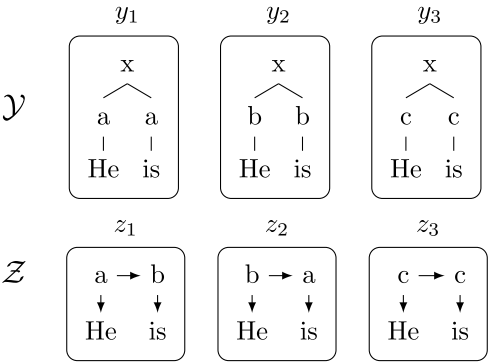
- $\mathcal{Q} \subseteq \mathcal{Q}'$

solutions:

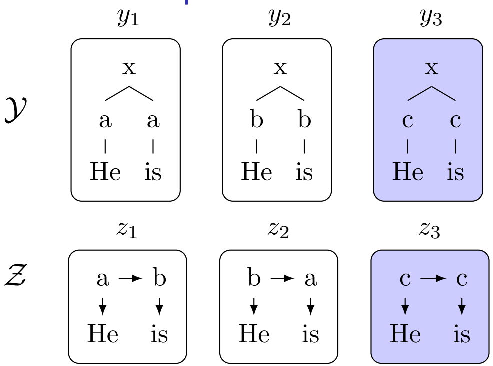
$$\max_{q \in \mathcal{Q}} h(q) \leq \max_{q \in \mathcal{Q}'} h(q) \text{ for any } h$$

Concrete example

- $\mathcal{Y} = \{y_1, y_2, y_3\}$
- $\mathcal{Z} = \{z_1, z_2, z_3\}$
- $\Delta_y \subset \mathbb{R}^3$, $\Delta_z \subset \mathbb{R}^3$



Simple solution



choose:

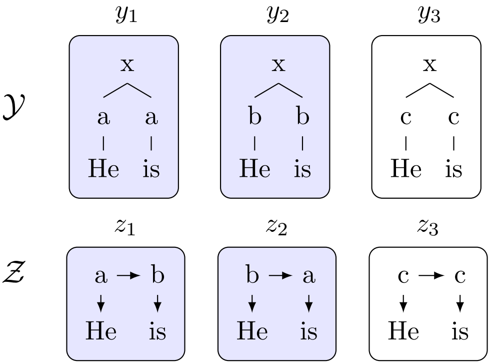
- $\alpha^{(1)} = (0,0,1) \in \Delta_y$ is representation of y_3
- $\beta^{(1)} = (0,0,1) \in \Delta_z$ is representation of z_3

confirm:

$$\sum_y \alpha_y^{(1)} y(i,t) = \sum_z \beta_z^{(1)} z(i,t)$$

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

Fractional solution



choose:

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

confirm:

$$\sum_{y} \alpha_y^{(2)} y(i,t) = \sum_{z} \beta_z^{(2)} z(i,t)$$

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

Optimal solution

weights:

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

example: $f = [1\ 1\ 2]$ and $g = [1\ 1\ -2]$

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

summary: dual and LP primal optimal:

$$\min_u M(u) = \min_u L(u) = \sum_y \alpha_y^{(2)} f(y) + \sum_z \beta_z^{(2)} g(z) = 2$$

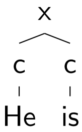
original primal optimal:

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

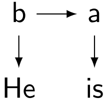
round 1

dual solutions:

y_3



z_2

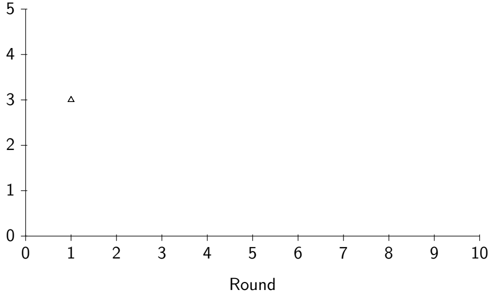


dual values:

$y^{(1)}$	2.00
$z^{(1)}$	1.00
$L(u^{(1)})$	3.00

previous solutions:

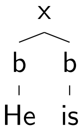
y_3 z_2



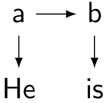
round 2

dual solutions:

y_2



z_1

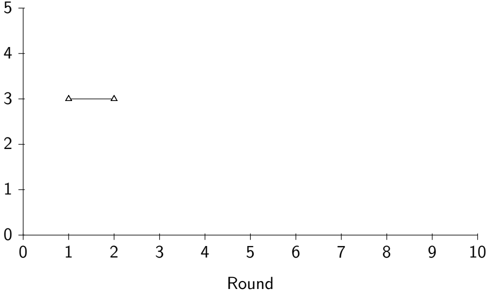


dual values:

$y^{(2)}$	2.00
$z^{(2)}$	1.00
$L(u^{(2)})$	3.00

previous solutions:

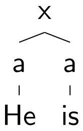
y_3	z_2
y_2	z_1



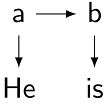
round 3

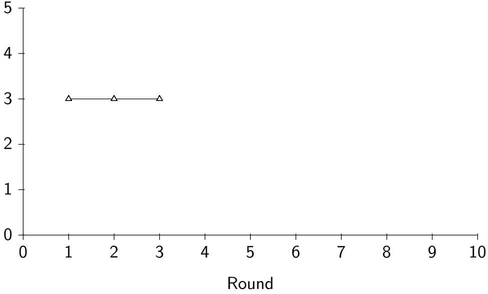
dual solutions:

y_1



z_1





dual values:

$y^{(3)}$	2.50
$z^{(3)}$	0.50
$L(u^{(3)})$	3.00

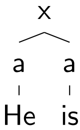
previous solutions:

y_3	z_2
y_2	z_1
y_1	z_1

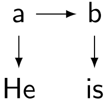
round 4

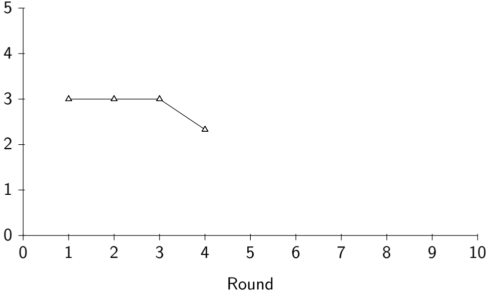
dual solutions:

y_1



z_1





dual values:

$y^{(4)}$	2.17
$z^{(4)}$	0.17
$L(u^{(4)})$	2.33

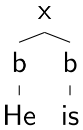
previous solutions:

y_3	z_2
y_2	z_1
y_1	z_1
y_1	z_1

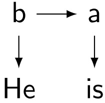
round 5

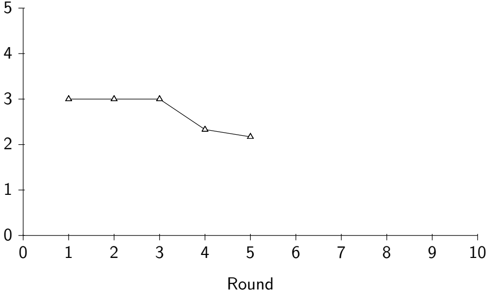
dual solutions:

y_2



z_2





dual values:

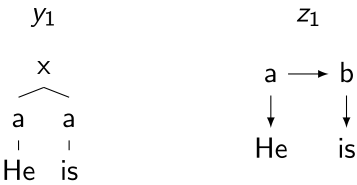
$y^{(5)}$	2.08
$z^{(5)}$	0.08
$L(u^{(5)})$	2.17

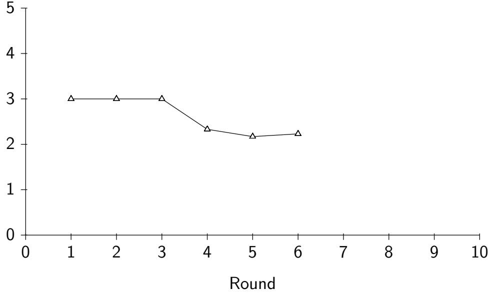
previous solutions:

y_3	z_2
y_2	z_1
y_1	z_1
y_1	z_1
y_2	z_2

round 6

dual solutions:





dual values:

$y^{(6)}$	2.12
$z^{(6)}$	0.12
$L(u^{(6)})$	2.23

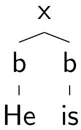
previous solutions:

y_3	z_2
y_2	z_1
y_1	z_1
y_1	z_1
y_2	z_2
y_1	z_1

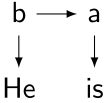
round 7

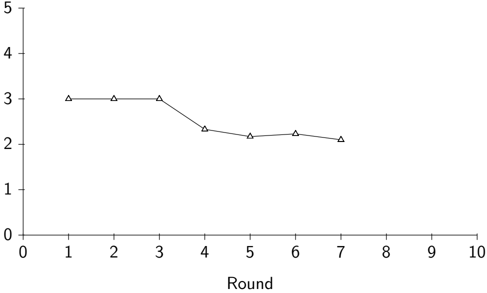
dual solutions:

y_2



z_2





dual values:

$y^{(7)}$	2.05
$z^{(7)}$	0.05
$L(u^{(7)})$	2.10

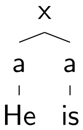
previous solutions:

y_3	z_2
y_2	z_1
y_1	z_1
y_1	z_1
y_2	z_2
y_1	z_1
y_2	z_2

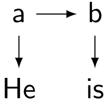
round 8

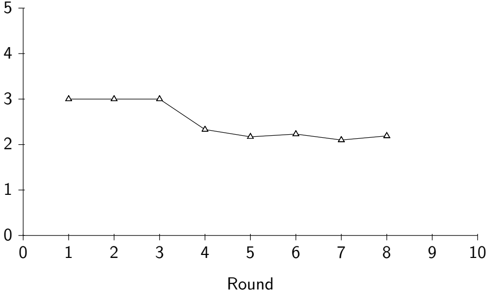
dual solutions:

y_1



z_1





dual values:

$y^{(8)}$	2.09
$z^{(8)}$	0.09
$L(u^{(8)})$	2.19

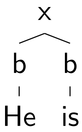
previous solutions:

y_3	z_2
y_2	z_1
y_1	z_1
y_1	z_1
y_2	z_2
y_1	z_1
y_2	z_2
y_1	z_1

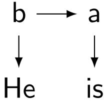
round 9

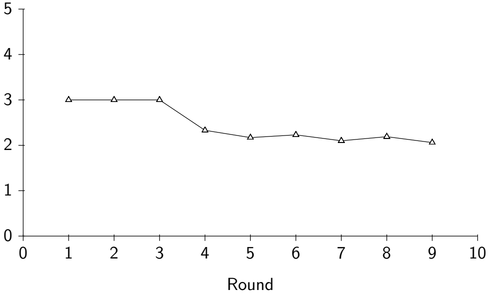
dual solutions:

y_2



z_2





dual values:

$y^{(9)}$	2.03
$z^{(9)}$	0.03
$L(u^{(9)})$	2.06

previous solutions:

y_3	z_2
y_2	z_1
y_1	z_1
y_1	z_1
y_2	z_2
y_1	z_1
y_2	z_2
y_1	z_1
y_2	z_2

Tighteningᅟ (Sherali and Adams, 1994; Sontag et al., 2008)

modify:

- extend $\mathcal{Y}$, $\mathcal{Z}$ to identify bigrams of part-of-speech tags
- $y(i, t_1, t_2) = 1 \leftrightarrow y(i, t_1) = 1$ and $y(i + 1, t_2) = 1$
- $z(i, t_1, t_2) = 1 \leftrightarrow z(i, t_1) = 1$ and $z(i + 1, t_2) = 1$

all bigram constraints: valid to add for all i, $t_1, t_2 \in \mathcal{T}$

$$\sum_y \alpha_y y(i, t_1, t_2) = \sum_z \beta_z z(i, t_1, t_2)$$

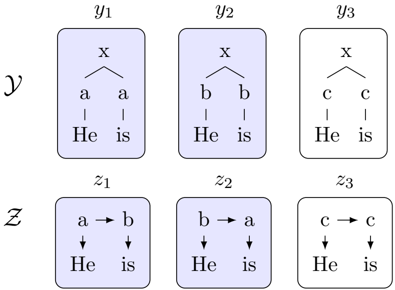
however this would make decoding expensive

Iterative tightening

single bigram constraint: cheaper to implement

$$\sum_{y} \alpha_y y(1, a, b) = \sum_{z} \beta_z z(1, a, b)$$

the solution $\alpha^{(1)}, \beta^{(1)}$ trivially passes this constraint, while $\alpha^{(2)}, \beta^{(2)}$ violates it



Dual decomposition with tightening

tightened decomposition includes an additional Lagrange multiplier

$$y_{u,v} = \arg\max_{y \in \mathcal{Y}} f(y) + \sum_{i,t} u(i,t)y(i,t) + v(1,a,b)y(1,a,b)$$

$$z_{u,v} = \arg\max_{z \in \mathcal{Z}} g(z) - \sum_{i,t} u(i,t)z(i,t) - v(1,a,b)z(1,a,b)$$

in general, this term can make the decoding problem more difficult

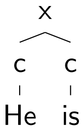
example:

- for small examples, these penalties are easy to compute
- for CFG parsing, need to include extra states that maintain tag bigrams (still faster than full intersection)

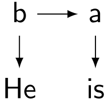
round 7

dual solutions:

y_3



z_2

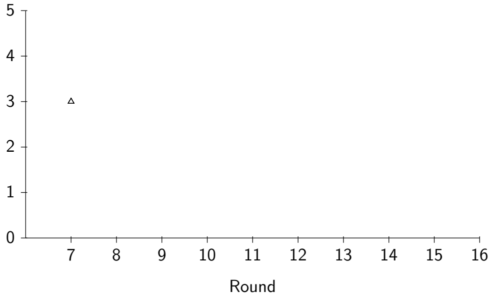


dual values:

$y^{(7)}$	2.00
$z^{(7)}$	1.00
$L(u^{(7)})$	3.00

previous solutions:

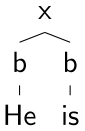
y_3 z_2



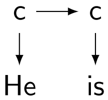
round 8

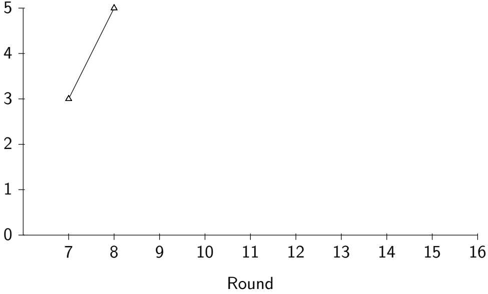
dual solutions:

y_2



z_3





dual values:

$y^{(8)}$	3.00
$z^{(8)}$	2.00
$L(u^{(8)})$	5.00

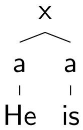
previous solutions:

y_3	z_2
y_2	z_3

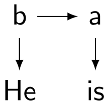
round 9

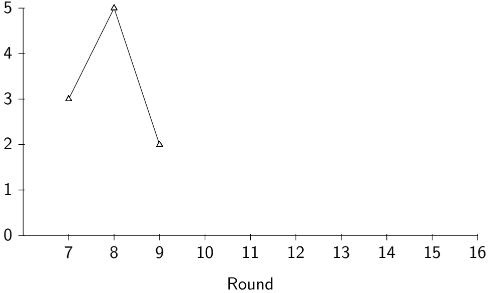
dual solutions:

y_1



z_2





dual values:

$y^{(9)}$	3.00
$z^{(9)}$	-1.00
$L(u^{(9)})$	2.00

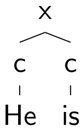
previous solutions:

y_3	z_2
y_2	z_3
y_1	z_2

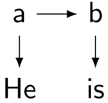
round 10

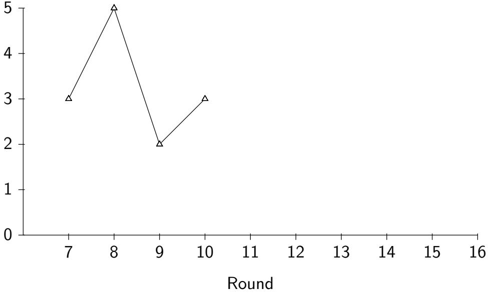
dual solutions:

y_3



z_1





dual values:

$y^{(10)}$	2.00
$z^{(10)}$	1.00
$L(u^{(10)})$	3.00

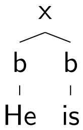
previous solutions:

y_3	z_2
y_2	z_3
y_1	z_2
y_3	z_1

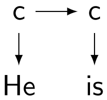
round 11

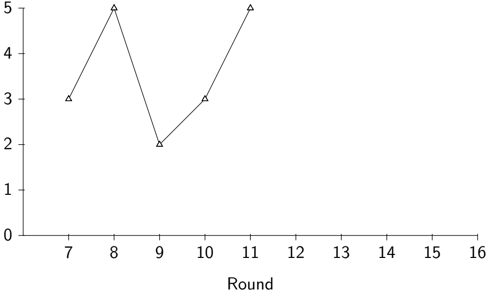
dual solutions:

y_2



z_3





dual values:

$y^{(11)}$	3.00
$z^{(11)}$	2.00
$L(u^{(11)})$	5.00

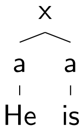
previous solutions:

y_3	z_2
y_2	z_3
y_1	z_2
y_3	z_1
y_2	z_3

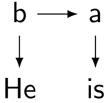
round 12

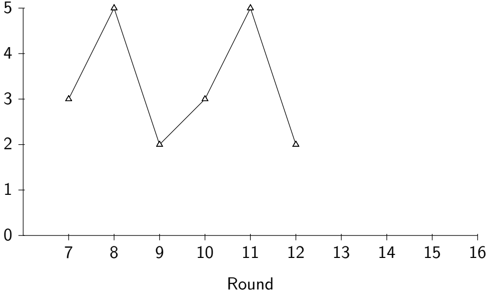
dual solutions:

y_1



z_2





dual values:

$y^{(12)}$	3.00
$z^{(12)}$	-1.00
$L(u^{(12)})$	2.00

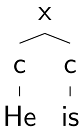
previous solutions:

y_3	z_2
y_2	z_3
y_1	z_2
y_3	z_1
y_2	z_3
y_1	z_2

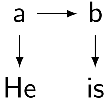
round 13

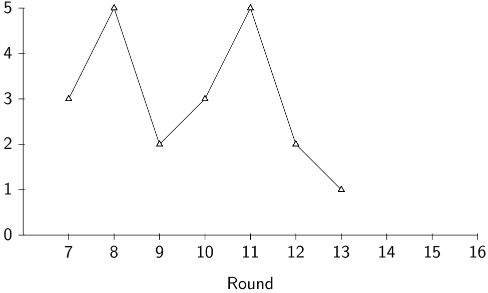
dual solutions:

y_3



z_1





dual values:

$y^{(13)}$	2.00
$z^{(13)}$	-1.00
$L(u^{(13)})$	1.00

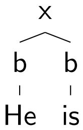
previous solutions:

y_3	z_2
y_2	z_3
y_1	z_2
y_3	z_1
y_2	z_3
y_1	z_2
y_3	z_1

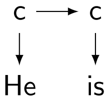
round 14

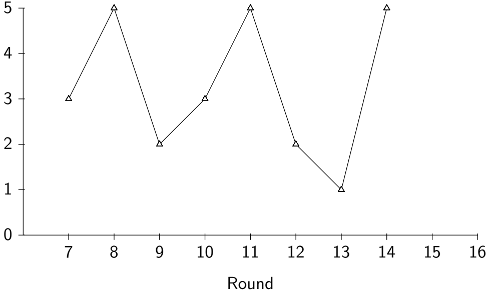
dual solutions:

y_2



z_3





dual values:

$y^{(14)}$	3.00
$z^{(14)}$	2.00
$L(u^{(14)})$	5.00

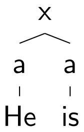
previous solutions:

y_3	z_2
y_2	z_3
y_1	z_2
y_3	z_1
y_2	z_3
y_1	z_2
y_3	z_1
y_2	z_3

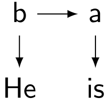
round 15

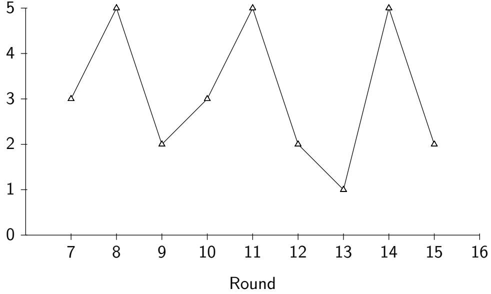
dual solutions:

y_1



z_2





dual values:

$y^{(15)}$	3.00
$z^{(15)}$	-1.00
$L(u^{(15)})$	2.00

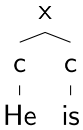
previous solutions:

y_3	z_2
y_2	z_3
y_1	z_2
y_3	z_1
y_2	z_3
y_1	z_2
y_3	z_1
y_2	z_3
y_1	z_2

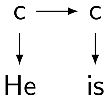
round 16

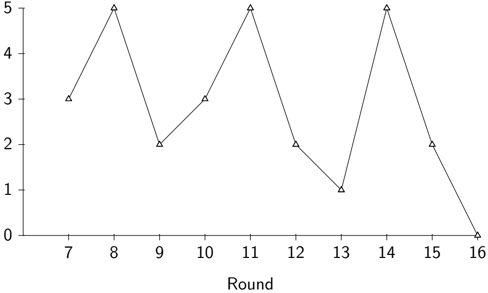
dual solutions:

y_3



z_3





dual values:

$y^{(16)}$	2.00
$z^{(16)}$	-2.00
$L(u^{(16)})$	0.00

previous solutions:

y_3	z_2
y_2	z_3
y_1	z_2
y_3	z_1
y_2	z_3
y_1	z_2
y_3	z_1
y_2	z_3
y_1	z_2
y_3	z_3

6. Advanced examples

aim: demonstrate some different relaxation techniques

- higher-order non-projective dependency parsing
- syntactic machine translation

Higher-order non-projective dependency parsing

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

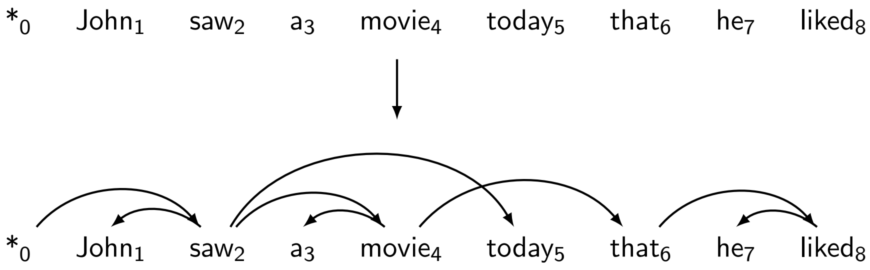
problem: find non-projective dependency parse that maximizes the score of this model

difficulty:

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

strategy: design a decomposition that separates combinatorial constraints from direct implementation of the scoring function

Non-Projective Dependency Parsing



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

$*_0$ 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

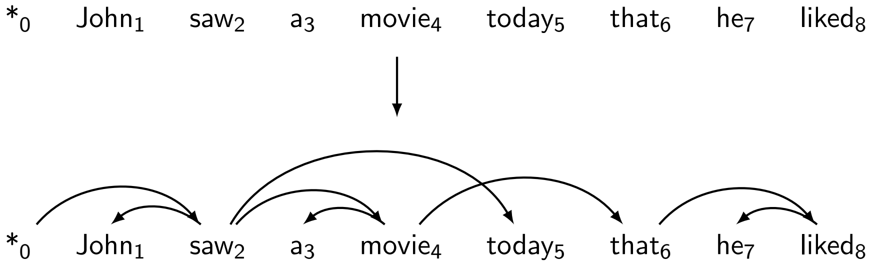
$$y^* = \arg\max_{y} f(y)$$

by decomposing into smaller problems.

Upshot: Can utilize a toolbox of combinatorial algorithms.

- Dynamic programming
- Minimum spanning tree
- Shortest path
- Min-Cut
- ...

Non-Projective Dependency Parsing

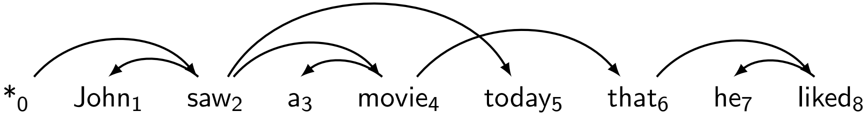


$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

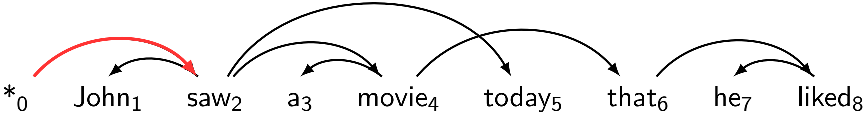
- Starts at the root symbol *
- Each word has a exactly one parent word
- Produces a tree structure (no cycles)
- Dependencies can cross

Arc-Factored



$f(y) =$

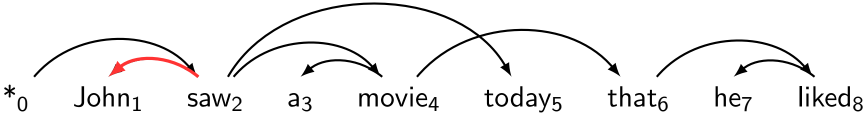
Arc-Factored



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

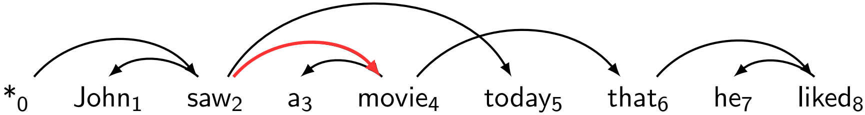
$$f(y) = \mathit{score}(\text{head } = *_0, \text{mod } = \text{saw}_2)$$

Arc-Factored



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

Arc-Factored

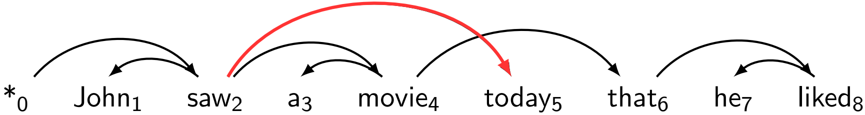


$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

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

Arc-Factored

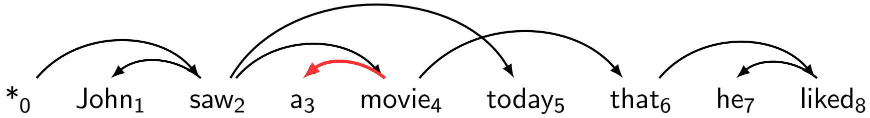


$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

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

Arc-Factored

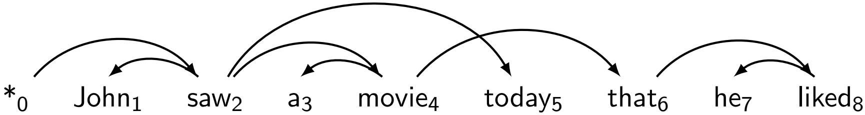


$$f(y) = \mathit{score}(\text{head} = *_0, \text{mod} = \text{saw}_2) + \mathit{score}(\text{saw}_2, \text{John}_1)$$

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

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

Arc-Factored



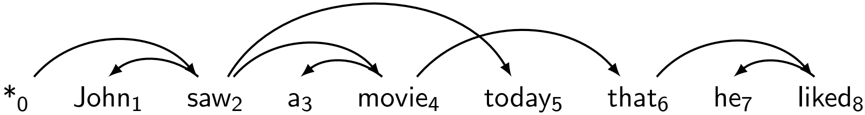
$$f(y) = \textit{score}(\mathrm{head} = *_0, \mathrm{mod} = \mathrm{saw}_2) + \textit{score}(\mathrm{saw}_2, \mathrm{John}_1)$$

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

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

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

Arc-Factored


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

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

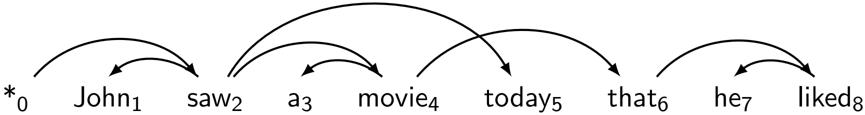
$$+\textit{score}(\text{saw}_2, \text{movie}_4) + \textit{score}(\text{saw}_2, \text{today}_5)$$

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

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)

Arc-Factored



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

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

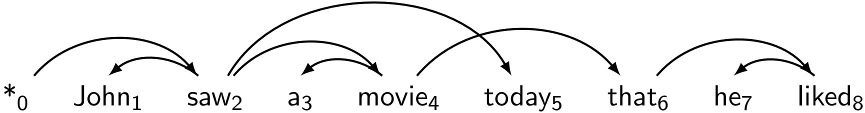
$$+\mathit{score}(\text{movie}_4, \text{a}_3) + \ldots$$

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

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

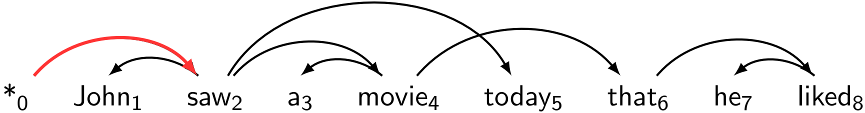
$y^* = \arg\max_y f(y) \Leftarrow$ Minimum Spanning Tree Algorithm

Sibling Models



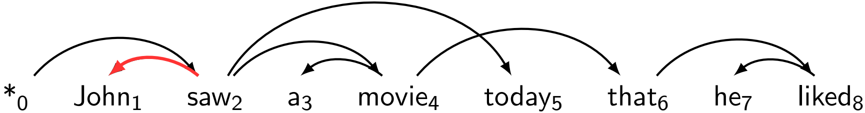
$f(y) =$

Sibling Models



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

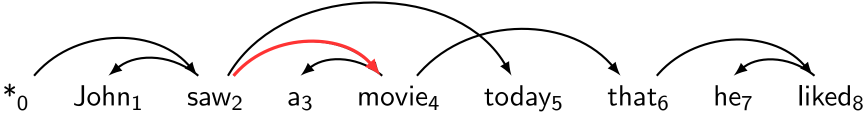
Sibling Models



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

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

Sibling Models

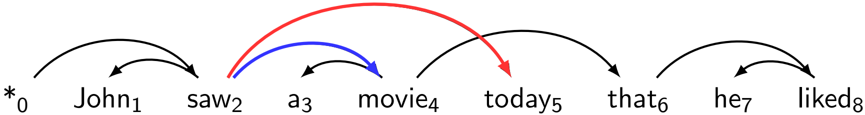


$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

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

Sibling Models

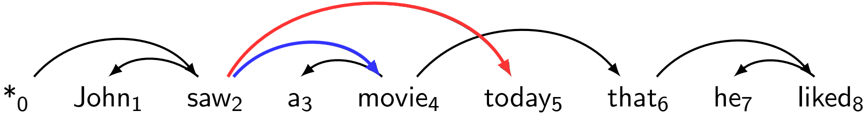


$$f(y) = \textit{score}(\textit{head} = *_0, \textit{prev} = \text{NULL}, \textit{mod} = \text{saw}_2)$$

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

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

Sibling Models



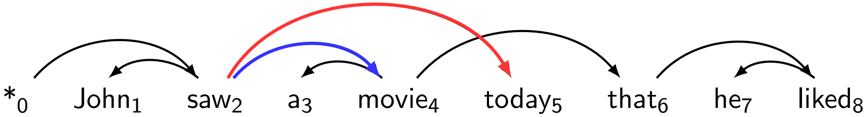
$$f(y) = score(head = *_0, prev = \text{NULL}, mod = \text{saw}_2)$$

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

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

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

Sibling Models



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

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

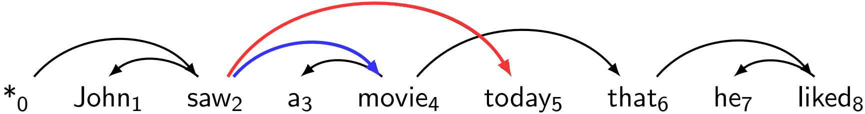
$$+\mathit{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \mathit{score}(\text{saw}_2, \text{NULL}, \text{movie}_4)$$

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

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

Sibling Models



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

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

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

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

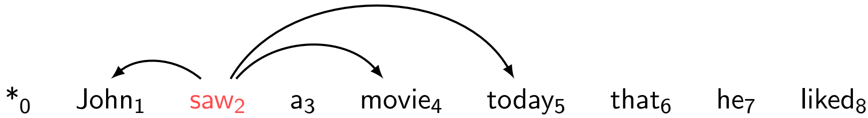
or $\mathit{score}(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5) = w \cdot \phi(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5)$

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

Thought Experiment: Individual Decoding

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

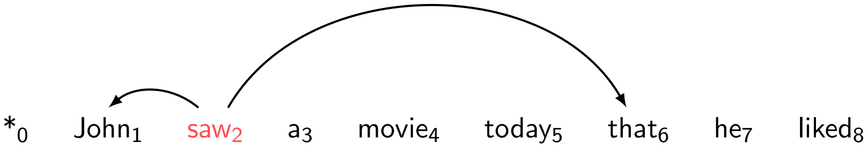
Thought Experiment: Individual Decoding



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

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

Thought Experiment: Individual Decoding

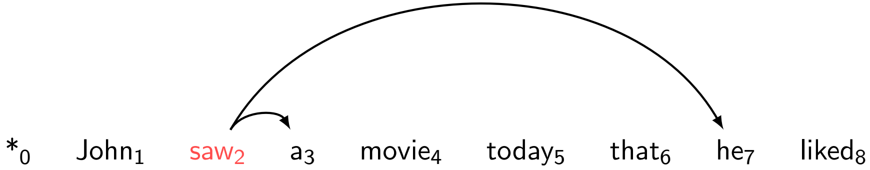


$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

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

Thought Experiment: Individual Decoding



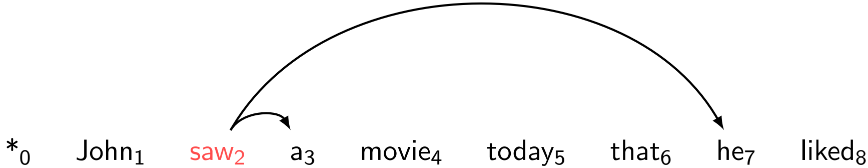
$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

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

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

Thought Experiment: Individual Decoding



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

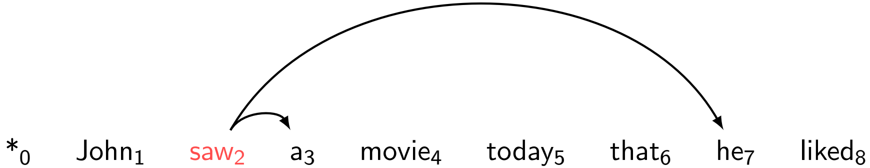
2^{n-1} **possibilities**

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

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

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

Thought Experiment: Individual Decoding



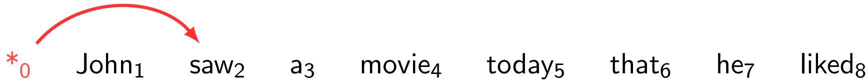
$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

$$\left\{\begin{array}{l} \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{movie}_4) \\ +\textit{score}(\mathrm{saw}_2, \mathrm{movie}_4, \mathrm{today}_5) \\ \hline \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{John}_1) + \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{that}_6) \\ \hline \textit{score}(\mathrm{saw}_2, \mathrm{NULL}, \mathrm{a}_3) + \textit{score}(\mathrm{saw}_2, \mathrm{a}_3, \mathrm{he}_7) \end{array}\right.$$

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

Thought Experiment Continued

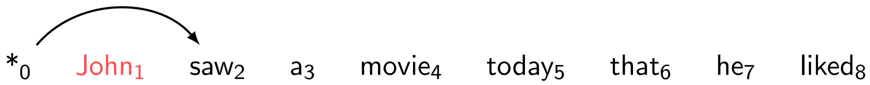


$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

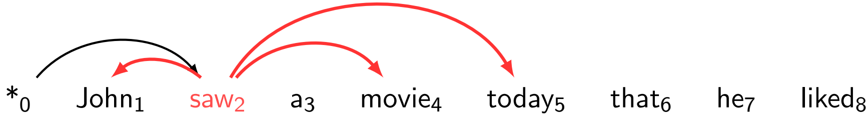


$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

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

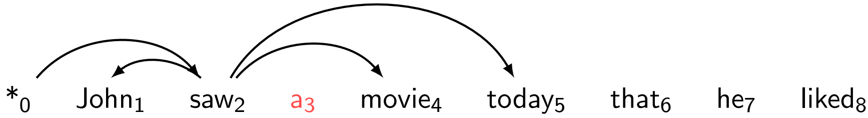
Thought Experiment Continued



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

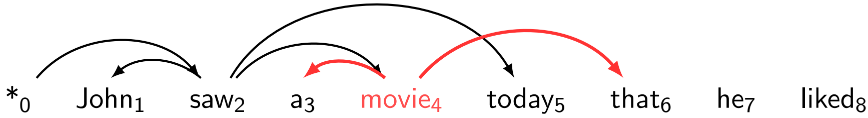


$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

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

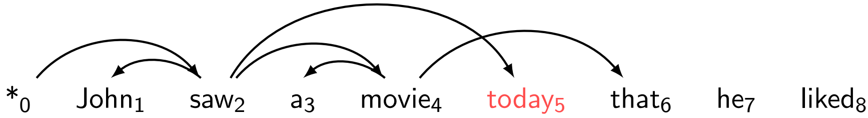
Thought Experiment Continued



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

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

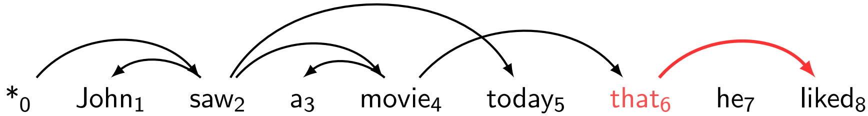
Thought Experiment Continued



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

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

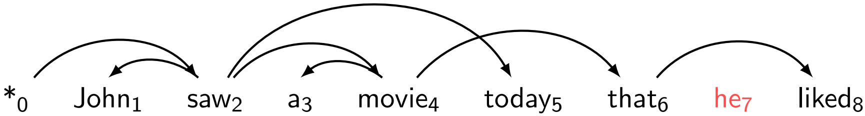
Thought Experiment Continued



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

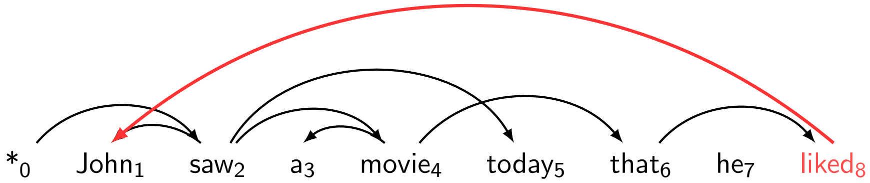


$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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



$*_0$ 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.

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

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

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

such that $z = y$

Dual Decomposition Structure

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

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

All Possible

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

Dual Decomposition Structure

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

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

All Possible

Valid Trees

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

Dual Decomposition Structure

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

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

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

Dual Decomposition Structure

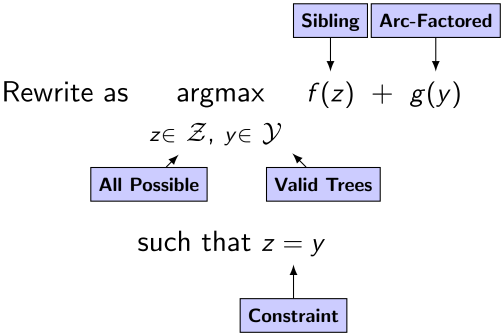
$$\text{Goal } y^* = \arg\max_{y \in \mathcal{Y}} f(y)$$

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

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

Dual Decomposition Structure

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



Sibling

Arc-Factored

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

All Possible

Valid Trees

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

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 $(f(z) + \mathrm{penalty})$ 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 $(y^{(k)}, z^{(k)})$

    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$_8$

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

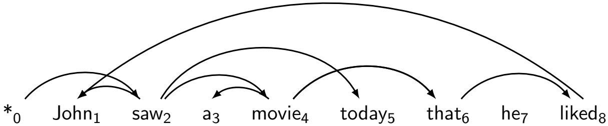
Key

$f(z)$	$\Leftarrow$	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



$*_0$ 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$_8$

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

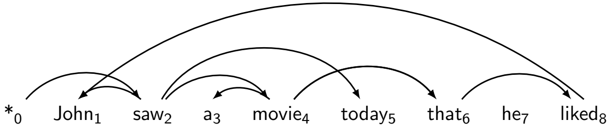
Key

$f(z)$	$\Leftarrow$	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

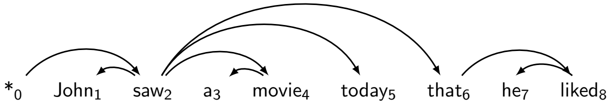
Penalties

$u(i,j) = 0$ 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



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

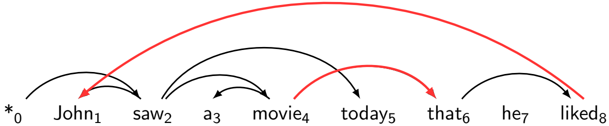
Key

$f(z)$	$\Leftarrow$	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

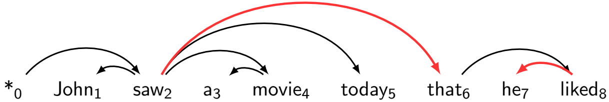
$u(i,j) = 0$ for all i,j



$*_0$ 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



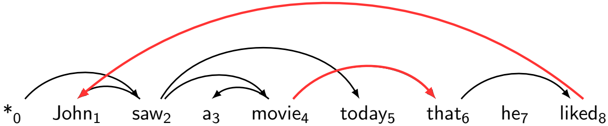
$*_0$ John$_1$ saw$_2$ a$_3$ movie$_4$ today$_5$ that$_6$ he$_7$ liked$_8$

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

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



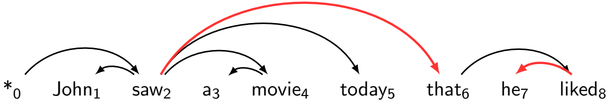
$$z^* = \arg\max_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j) z(i,j))$$

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

Minimum Spanning Tree

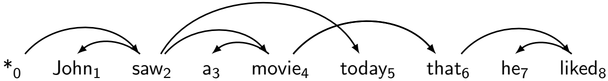


$$y^* = \arg\max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j) y(i,j))$$

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



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

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

Minimum Spanning Tree

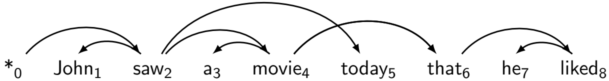
$*_0$ John_1 saw_2 a_3 movie_4 today_5 that_6 he_7 liked_8

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

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



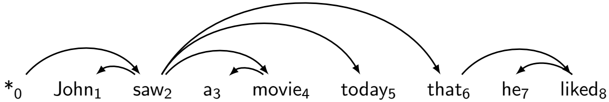
$$z^* = \arg\max_{z \in \mathcal{Z}} (f(z) + \sum_{i,j} u(i,j) z(i,j))$$

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

Minimum Spanning Tree

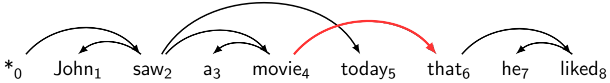


$$y^* = \arg\max_{y \in \mathcal{Y}} (g(y) - \sum_{i,j} u(i,j) y(i,j))$$

Key

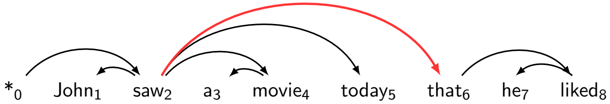
$f(z)$	$\Leftarrow$	Sibling Model	$g(y)$	$\Leftarrow$	Arc-Factored Model
$\mathcal{Z}$	$\Leftarrow$	No Constraints	$\mathcal{Y}$	$\Leftarrow$	Tree Constraints
$y(i,j) = 1$	if	y contains dependency i,j			

Individual Decoding


$*_0$ 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

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

$*_0$ 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

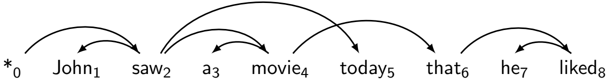
Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

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

$*_0$ 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

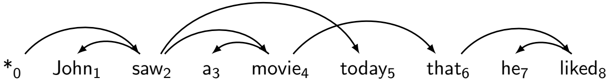
Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

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

Key

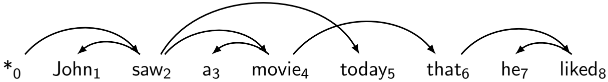
$f(z)$	⇐	Sibling Model	$g(y)$	⇐	Arc-Factored Model
$\mathcal{Z}$	⇐	No Constraints	$\mathcal{Y}$	⇐	Tree Constraints
$y(i,j) = 1$	if	y contains dependency i,j			

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

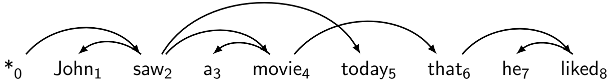
Iteration 1	
$u(8,1)$	-1
$u(4,6)$	-1
$u(2,6)$	1
$u(8,7)$	1

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

Key

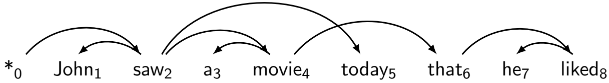
$f(z)$	$\Leftarrow$	Sibling Model	$g(y)$	$\Leftarrow$	Arc-Factored Model
$\mathcal{Z}$	$\Leftarrow$	No Constraints	$\mathcal{Y}$	$\Leftarrow$	Tree Constraints
$y(i,j) = 1$	if	y contains dependency i,j			

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

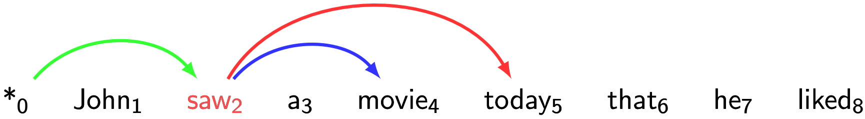
If at any iteration $y^{(k)} = z^{(k)}$, then $(y^{(k)}, z^{(k)})$ is the global optimum.

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



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

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

- Head Automata (Eisner, 2000)

  Generalization of Sibling models

  Allow arbitrary automata as local scoring function.

Experiments

Properties:

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

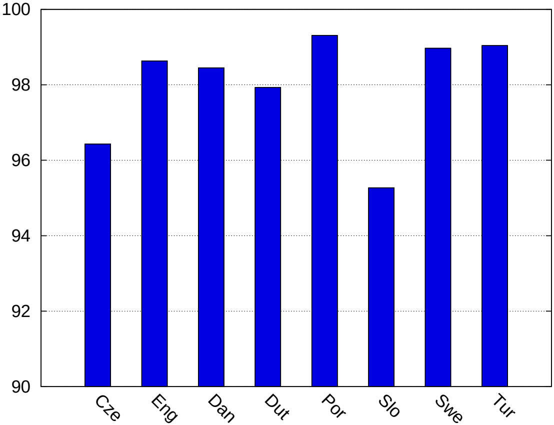
Training:

- Averaged Perceptron (more details in paper)

Experiments on:

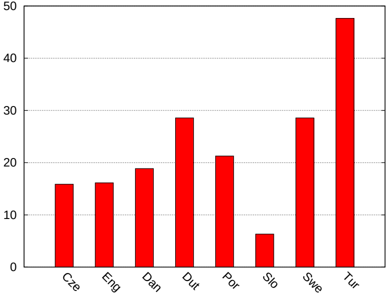
- CoNLL Datasets
- English Penn Treebank
- Czech Dependency Treebank

How often do we exactly solve the problem?

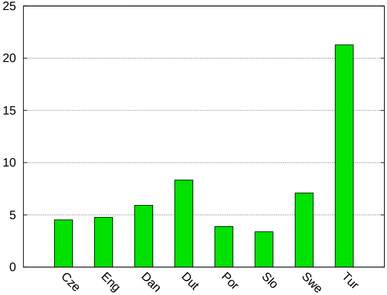


- Percentage of examples where the dual decomposition finds an exact solution.

Parsing Speed



Sibling model



Grandparent model

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

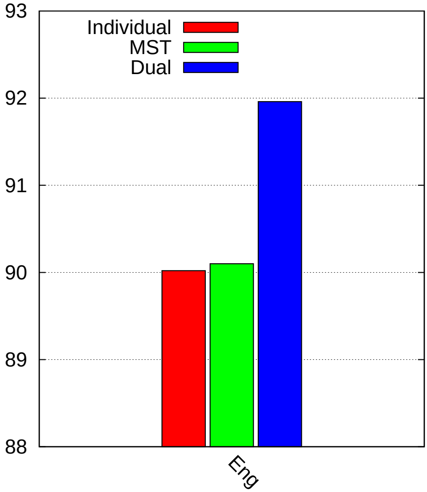
Accuracy

	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

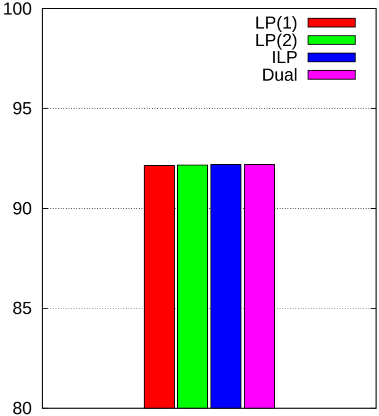
Use an LP/ILP Solver for decoding

We compare:

- Accuracy
- Exactness
- Speed

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

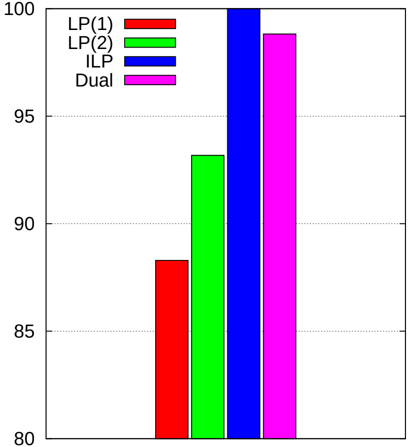
Comparison to LP/ILP: Accuracy



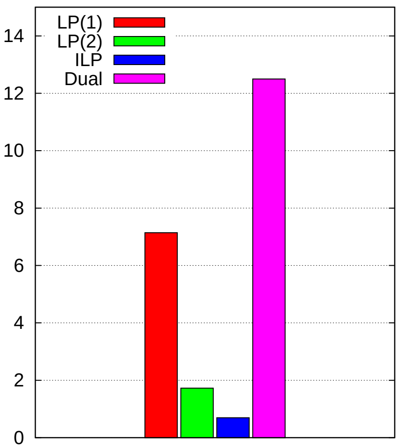
Dependency Accuracy

- All decoding methods have comparable accuracy

Comparison to LP/ILP: Exactness and Speed



Percentage with exact solution



Sentences per second

Syntactic translation decoding

setup: assume a trained model for syntactic machine translation

problem: find best derivation that maximizes the score of this model

difficulty:

- need to incorporate language model in decoding
- empirically, relaxation is often not tight, so dual decomposition does not always converge

strategy:

- use a different relaxation to handle language model
- incrementally add constraints to find exact solution

Syntactic Translation

Problem:

Decoding synchronous grammar for machine translation

Example:

<s> abarks le dug </s>

↓

<s> the dog barks loudly </s>

Goal:

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

where y is a parse derivation in a synchronous grammar

Hiero Example

Consider the input sentence

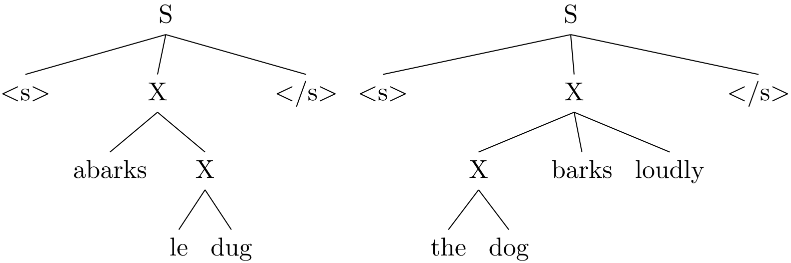
<s> abarks le dug </s>

And the synchronous grammar

S → <s> X </s>, <s> X </s>
X → abarks X, X barks loudly
X → abarks X, barks X
X → abarks X, barks X loudly
X → le dug, the dog
X → le dug, a cat

Hiero Example

Apply synchronous rules to map this sentence



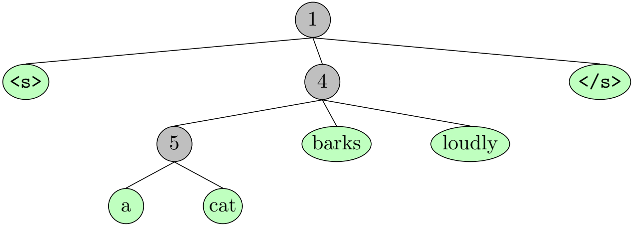
Many possible mappings:

```
<s> the dog barks loudly </s>
<s> a cat barks loudly </s>
<s> barks the dog </s>
<s> barks a cat </s>
<s> barks the dog loudly </s>
<s> barks a cat loudly </s>
```

Translation Forest

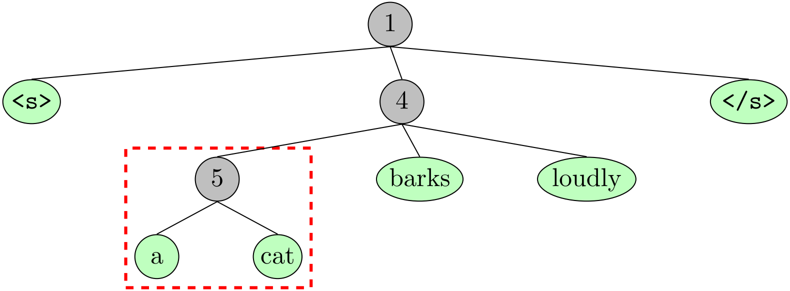
Rule	Score
1 → `<s>` 4 `</s>`	-1
4 → 5 barks loudly	2
4 → barks 5	0.5
4 → barks 5 loudly	3
5 → the dog	-4
5 → a cat	2.5

Example: a derivation in the translation forest



Scoring function

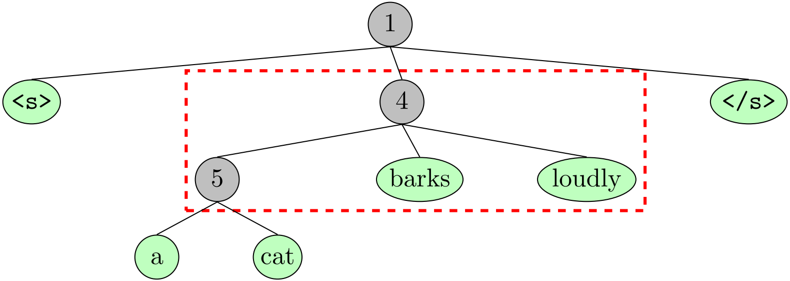
Score : sum of hypergraph derivation and language model



$f(y) = \mathit{score}(5 \rightarrow \text{a cat})$

Scoring function

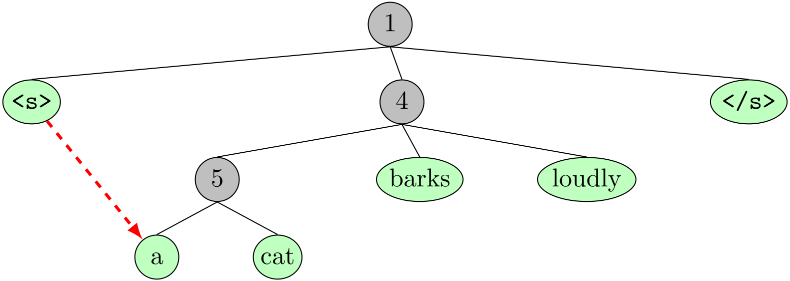
Score : sum of hypergraph derivation and language model



$f(y) = \mathit{score}(5 \rightarrow \text{a cat}) + \mathit{score}(4 \rightarrow 5 \text{ barks loudly})$

Scoring function

Score : sum of hypergraph derivation and language model

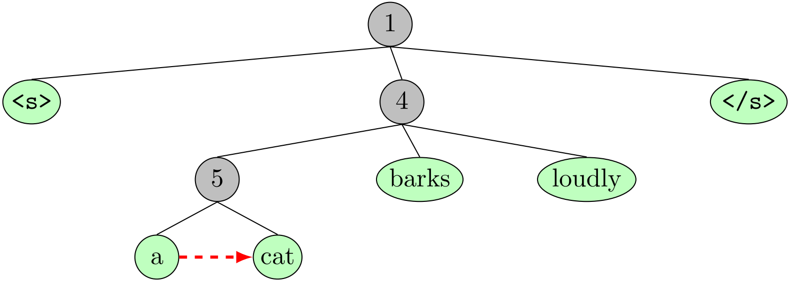


$$f(y) = \textit{score}(5 \to \text{a cat}) + \textit{score}(4 \to \text{5 barks loudly}) + \ldots$$

$$+\textit{score}(\texttt{<s>}, \text{the})$$

Scoring function

Score : sum of hypergraph derivation and language model

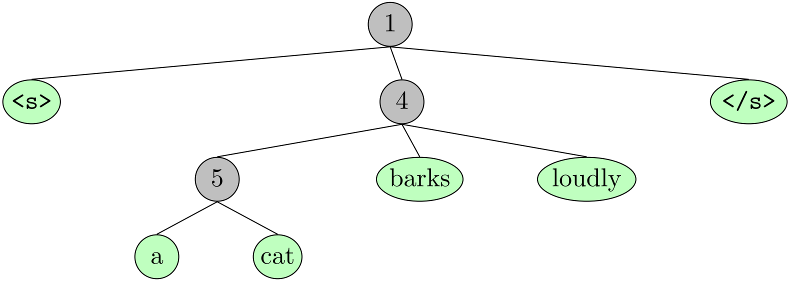


$$f(y) = \mathit{score}(5 \rightarrow \text{a cat}) + \mathit{score}(4 \rightarrow 5 \text{ barks loudly}) + \ldots$$

$$+\mathit{score}(\texttt{<s>}, \text{a}) + \mathit{score}(\text{a}, \text{cat})$$

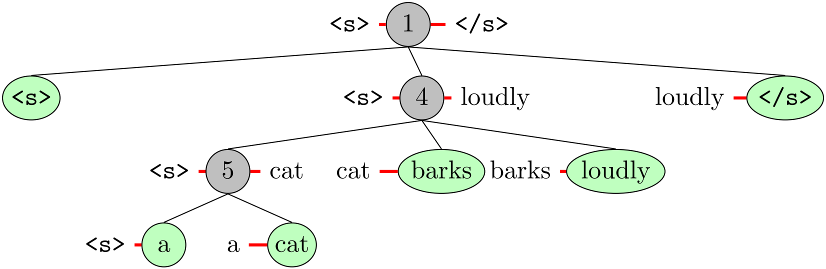
Exact Dynamic Programming

To maximize combined model, need to ensure that bigrams are consistent with parse tree.



Exact Dynamic Programming

To maximize combined model, need to ensure that bigrams are consistent with parse tree.



Original Rules
$5 \rightarrow \text{the} \;\; \text{dog}$
$5 \rightarrow \text{a} \;\; \text{cat}$

New Rules
${}_{<s>}5_{cat} \rightarrow {}_{<s>}the_{the}\ {}_{the}dog_{dog}$
${}_{barks}5_{cat} \rightarrow {}_{barks}the_{the}\ {}_{the}dog_{dog}$
${}_{<s>}5_{cat} \rightarrow {}_{<s>}a_{a}\ {}_{a}cat_{cat}$
${}_{barks}5_{cat} \rightarrow {}_{barks}a_{a}\ {}_{a}cat_{cat}$

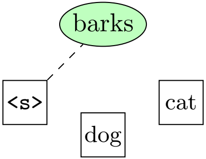
Lagrangian Relaxation Algorithm for Syntactic Translation

Outline:

- Algorithm for simplified version of translation
- Full algorithm with certificate of exactness
- Experimental results

Thought experiment: Greedy language model

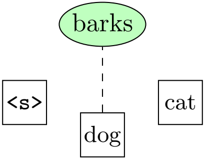
Choose best bigram for a given word



- *score*(<s>, barks)

Thought experiment: Greedy language model

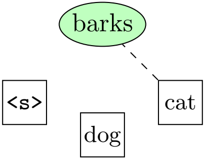
Choose best bigram for a given word



- *score*(<s>, barks)
- *score*(dog, barks)

Thought experiment: Greedy language model

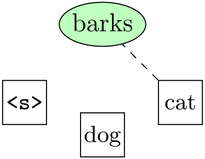
Choose best bigram for a given word



- *score*(<s>, barks)
- *score*(dog, barks)
- *score*(cat, barks)

Thought experiment: Greedy language model

Choose best bigram for a given word



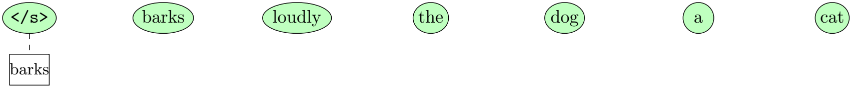
- *score*(<s>, barks)
- *score*(dog, barks)
- *score*(cat, barks)

Can compute with a simple maximization

$$\arg \max_{w : \langle w, \text{barks} \rangle \in \mathcal{B}} \textit{score}(w, \text{barks})$$

Thought experiment: Full decoding

Step 1. Greedily choose best bigram for each word

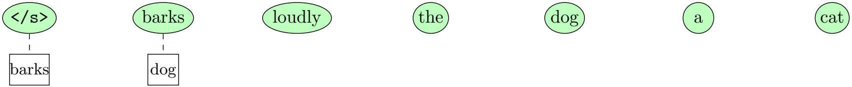


</s> barks loudly the dog a cat

barks

Thought experiment: Full decoding

Step 1. Greedily choose best bigram for each word

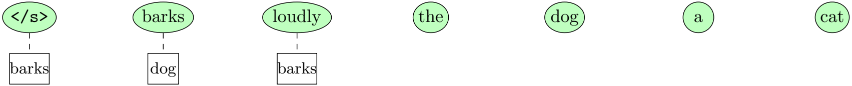


</s> barks loudly the dog a cat

barks dog

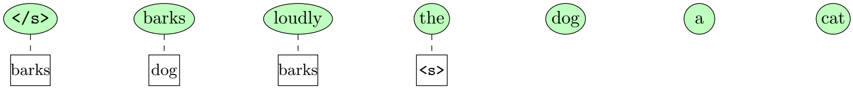
Thought experiment: Full decoding

Step 1. Greedily choose best bigram for each word



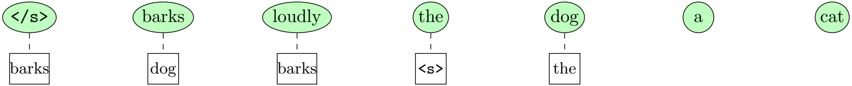
Thought experiment: Full decoding

Step 1. Greedily choose best bigram for each word



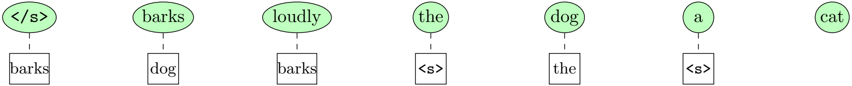
Thought experiment: Full decoding

Step 1. Greedily choose best bigram for each word



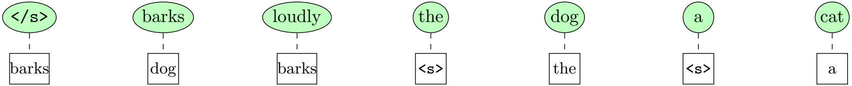
Thought experiment: Full decoding

Step 1. Greedily choose best bigram for each word



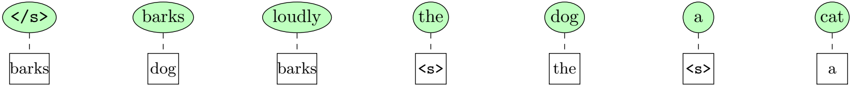
Thought experiment: Full decoding

Step 1. Greedily choose best bigram for each word



Thought experiment: Full decoding

Step 1. Greedily choose best bigram for each word



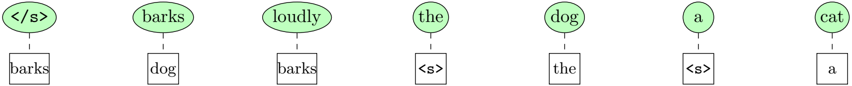
</s> barks loudly the dog a cat

barks dog barks <s> the <s> a

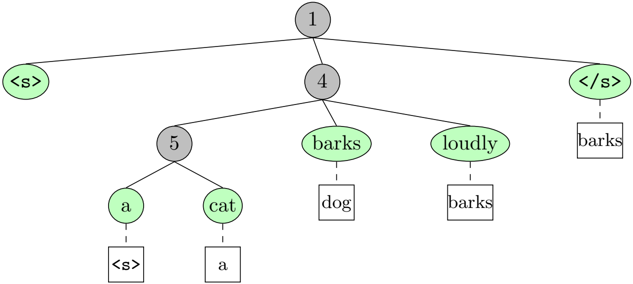
Step 2. Find the best derivation with fixed bigrams

Thought experiment: Full decoding

Step 1. Greedily choose best bigram for each word

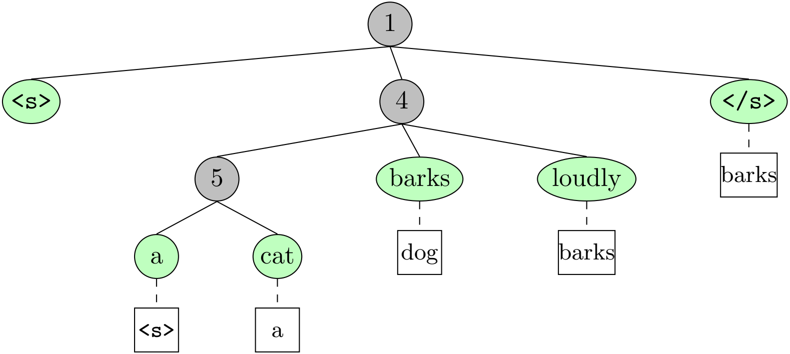


Step 2. Find the best derivation with fixed bigrams



Thought Experiment Problem

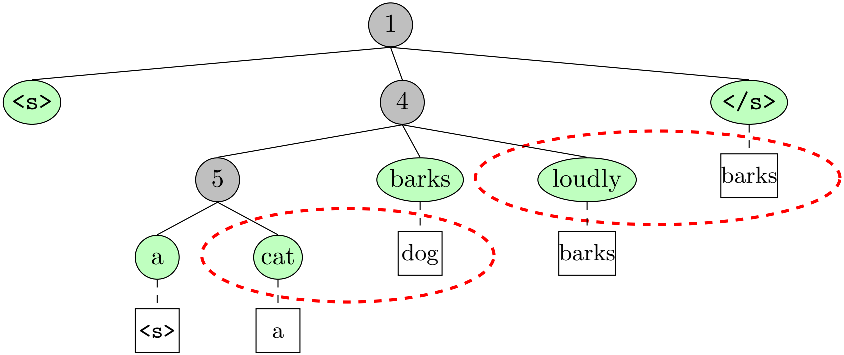
May produce invalid parse and bigram relationship



Greedy bigram selection may conflict with the parse derivation

Thought Experiment Problem

May produce invalid parse and bigram relationship



Greedy bigram selection may conflict with the parse derivation

Formal objective

Notation: $y(w, v) = 1$ if the bigram $\langle w, v \rangle \in \mathcal{B}$ is in y

Goal:

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

such that for all words nodes y_v

(1)

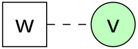
Formal objective

Notation: $y(w,v) = 1$ if the bigram $\langle w, v\rangle \in \mathcal{B}$ is in y

Goal:

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

such that for all words nodes y_v



$$y_v = \sum_{w:\langle w,v\rangle\in\mathcal{B}} y(w,v) \tag{1}$$

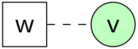
Formal objective

Notation: $y(w, v) = 1$ if the bigram $\langle w, v\rangle \in \mathcal{B}$ is in y

Goal:

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

such that for all words nodes y_v



$$y_v = \sum_{w : \langle w, v\rangle \in \mathcal{B}} y(w, v) \qquad (1)$$

$$y_v = \sum_{w : \langle v, w\rangle \in \mathcal{B}} y(v, w) \qquad (2)$$

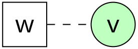
Formal objective

Notation: $y(w, v) = 1$ if the bigram $\langle w, v \rangle \in \mathcal{B}$ is in y

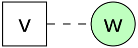
Goal:

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

such that for all words nodes y_v



$$y_v = \sum_{w : \langle w, v \rangle \in \mathcal{B}} y(w, v) \tag{1}$$



$$y_v = \sum_{w : \langle v, w \rangle \in \mathcal{B}} y(v, w) \tag{2}$$

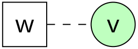
Formal objective

Notation: $y(w, v) = 1$ if the bigram $\langle w, v \rangle \in \mathcal{B}$ is in y

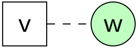
Goal:

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

such that for all words nodes y_v



$$y_v = \sum_{w:\langle w,v \rangle \in \mathcal{B}} y(w, v) \tag{1}$$



$$y_v = \sum_{w:\langle v,w \rangle \in \mathcal{B}} y(v, w) \tag{2}$$

Lagrangian: Relax constraint (2), leave constraint (1)

$$L(u, y) = \max_{y \in \mathcal{Y}} f(y) + \sum_{w,v} u(v) \left(y_v - \sum_{w:\langle v,w \rangle \in \mathcal{B}} y(v, w) \right)$$

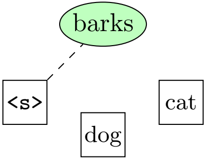
For a given u, $L(u, y)$ can be solved by our greedy LM algorithm

Algorithm

```
Set $u^{(1)}(v) = 0$ for all $v \in V_L$

For $k = 1$ to $K$

    $y^{(k)} \leftarrow \arg\max_{y \in \mathcal{Y}} L^{(k)}(u, y)$

    If $y_v^{(k)} = \sum_{w:\langle v,w \rangle \in \mathcal{B}} y^{(k)}(v, w)$ for all $v$ Return $(y^{(k)})$

    Else
        $u^{(k+1)}(v) \leftarrow u^{(k)}(v) - \alpha_k \left( y_v^{(k)} - \sum_{w:\langle v,w \rangle \in \mathcal{B}} y^{(k)}(v, w) \right)$
```

Thought experiment: Greedy with penalties

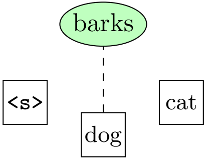
Choose best bigram with penalty for a given word



- $score(\texttt{<s>}, \text{barks}) - u(\texttt{<s>}) + u(\text{barks})$

Thought experiment: Greedy with penalties

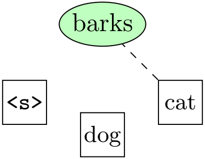
Choose best bigram with penalty for a given word



- $\textit{score}(\texttt{<s>}, \text{barks}) - u(\texttt{<s>}) + u(\text{barks})$
- $\textit{score}(\text{cat}, \text{barks}) - u(\text{cat}) + u(\text{barks})$

Thought experiment: Greedy with penalties

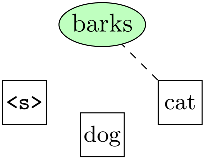
Choose best bigram with penalty for a given word



- $\mathit{score}(\texttt{<s>}, \text{barks}) - u(\texttt{<s>}) + u(\text{barks})$
- $\mathit{score}(\text{cat}, \text{barks}) - u(\text{cat}) + u(\text{barks})$
- $\mathit{score}(\text{dog}, \text{barks}) - u(\text{dog}) + u(\text{barks})$

Thought experiment: Greedy with penalties

Choose best bigram with penalty for a given word



- $\mathit{score}(\texttt{<s>}, \text{barks}) - u(\texttt{<s>}) + u(\text{barks})$
- $\mathit{score}(\text{cat}, \text{barks}) - u(\text{cat}) + u(\text{barks})$
- $\mathit{score}(\text{dog}, \text{barks}) - u(\text{dog}) + u(\text{barks})$

Can still compute with a simple maximization over

$$\arg \max_{w : \langle w, \text{barks} \rangle \in \mathcal{B}} \mathit{score}(w, \text{barks}) - u(w) + u(\text{barks})$$

Algorithm example

Penalties

v	`</s>`	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	0	0	0	0	0	0

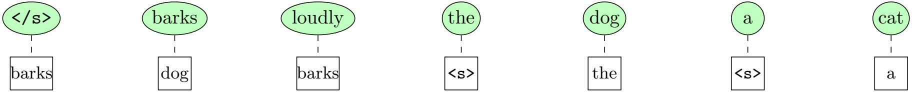
Greedy decoding

Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	0	0	0	0	0	0

Greedy decoding

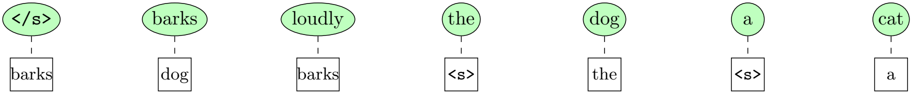


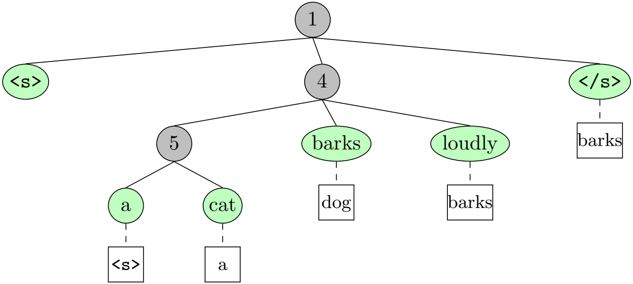
Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	0	0	0	0	0	0

Greedy decoding



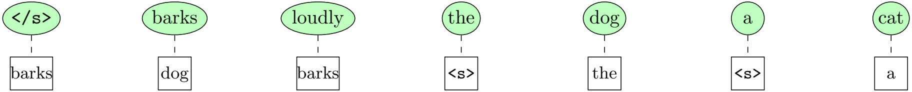


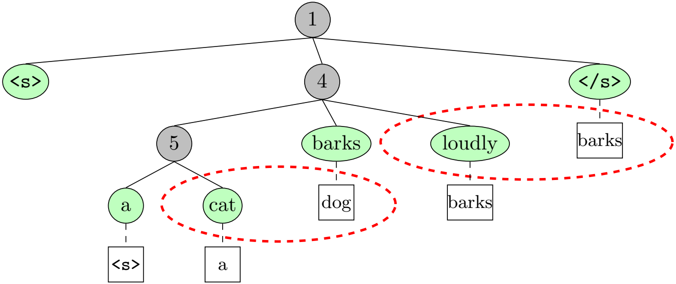
Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	0	0	0	0	0	0

Greedy decoding



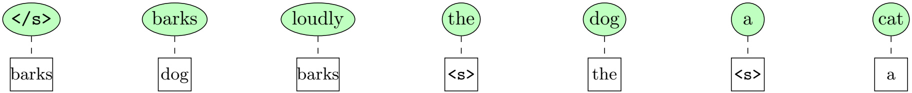


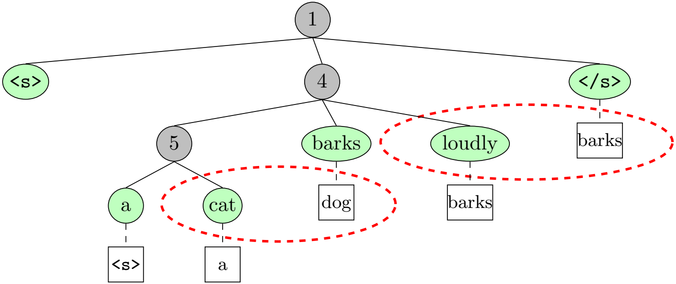
Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	-1	1	0	-1	0	1

Greedy decoding





Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	-1	1	0	-1	0	1

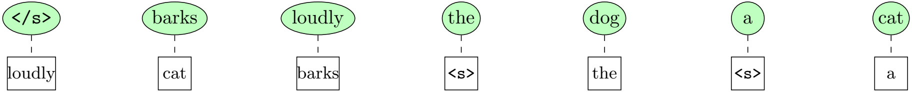
Greedy decoding

Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	-1	1	0	-1	0	1

Greedy decoding



</s> loudly

barks cat

loudly barks

the <s>

dog the

a <s>

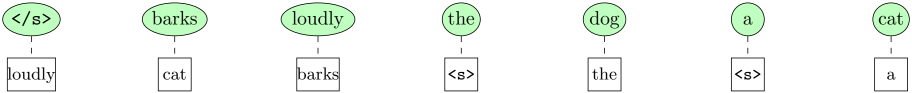
cat a

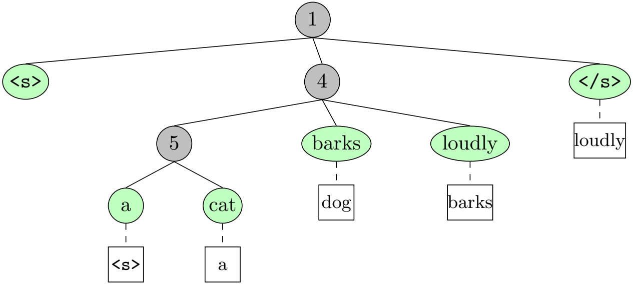
Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	-1	1	0	-1	0	1

Greedy decoding



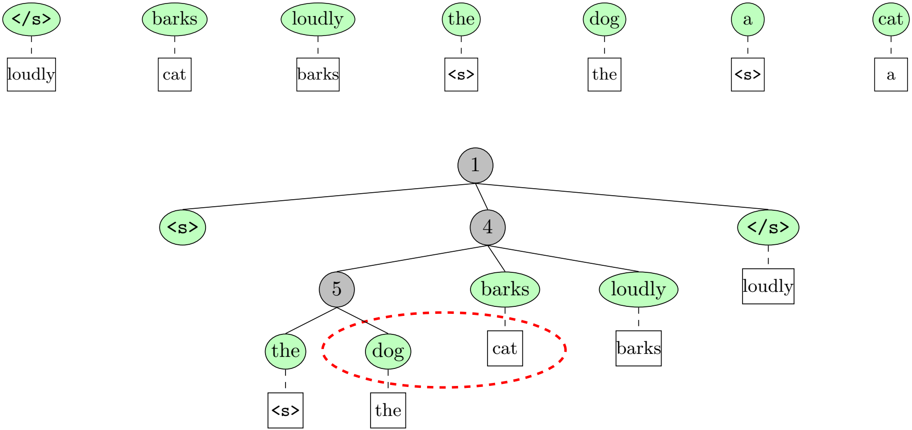


Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	-1	1	0	-1	0	1

Greedy decoding

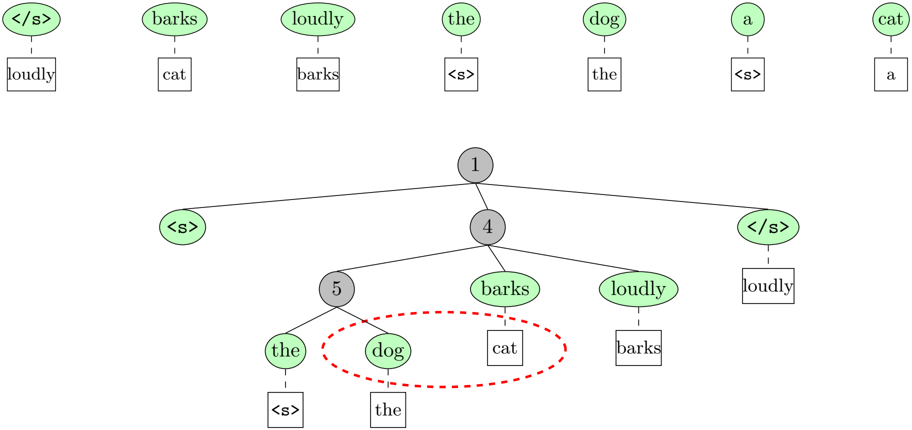


Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	-1	1	0	-0.5	0	0.5

Greedy decoding



Algorithm example

Penalties

v	`</s>`	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	-1	1	0	-0.5	0	0.5

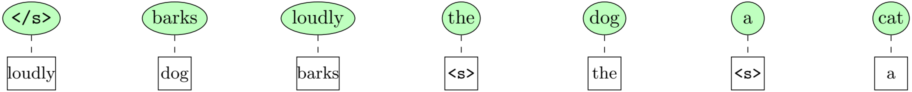
Greedy decoding

Algorithm example

Penalties

v	</s>	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	-1	1	0	-0.5	0	0.5

Greedy decoding

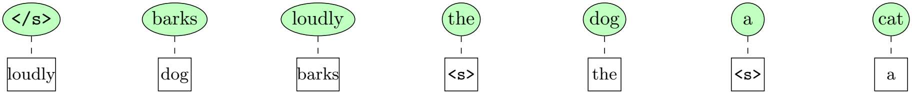


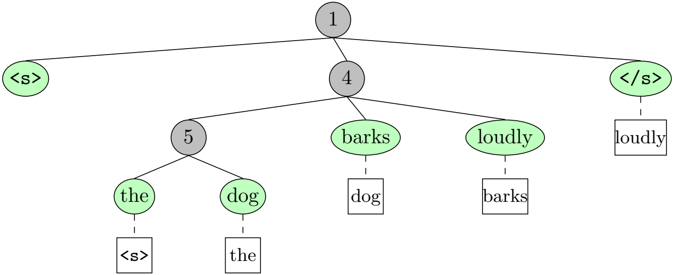
Algorithm example

Penalties

v	*</s>*	*barks*	*loudly*	*the*	*dog*	*a*	*cat*
u(v)	0	-1	1	0	-0.5	0	0.5

Greedy decoding





Constraint Issue

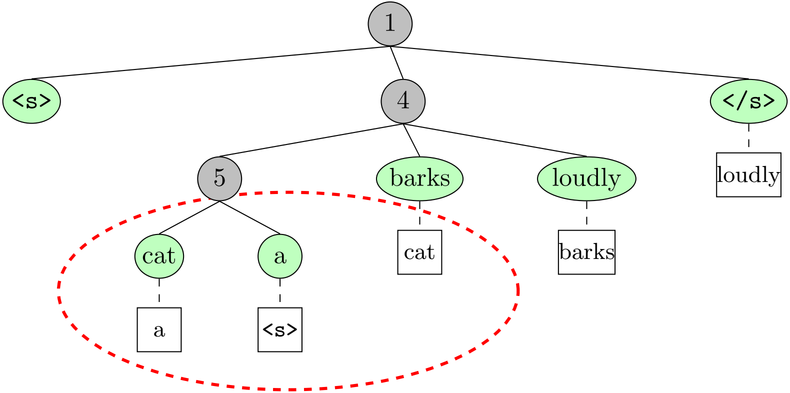
Constraints do not capture all possible reorderings

Example: Add rule $\langle 5 \rightarrow \text{cat a} \rangle$ to forest. New derivation

Constraint Issue

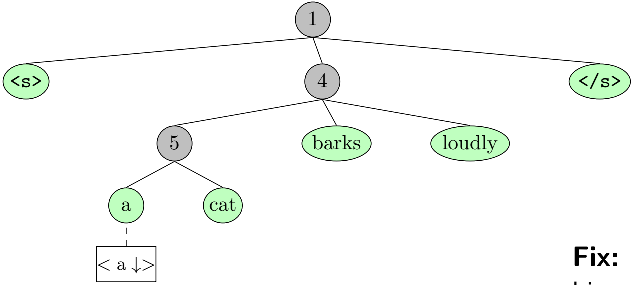
Constraints do not capture all possible reorderings

Example: Add rule ⟨5 → cat a⟩ to forest. New derivation



Satisfies both constraints (1) and (2), but is not self-consistent.

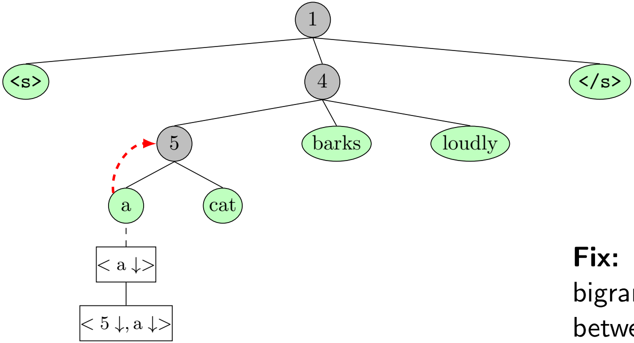
New Constraints: Paths



Fix: In addition to bigrams, consider paths between terminal nodes

Example: Path marker $\langle 5 \downarrow, 10 \downarrow \rangle$ implies that between two word nodes, we move down from node 5 to node 10

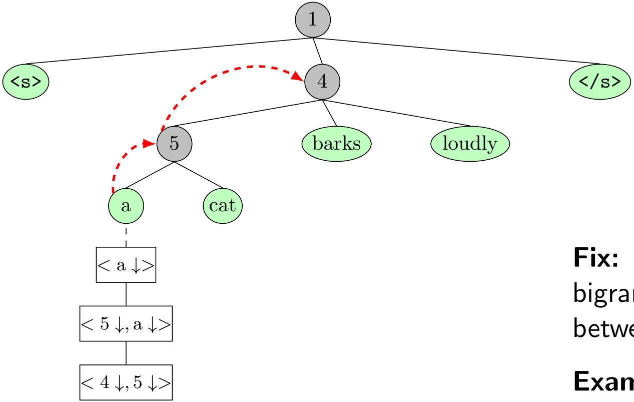
New Constraints: Paths



Fix: In addition to bigrams, consider paths between terminal nodes

Example: Path marker $\langle 5 \downarrow, 10 \downarrow \rangle$ implies that between two word nodes, we move down from node 5 to node 10

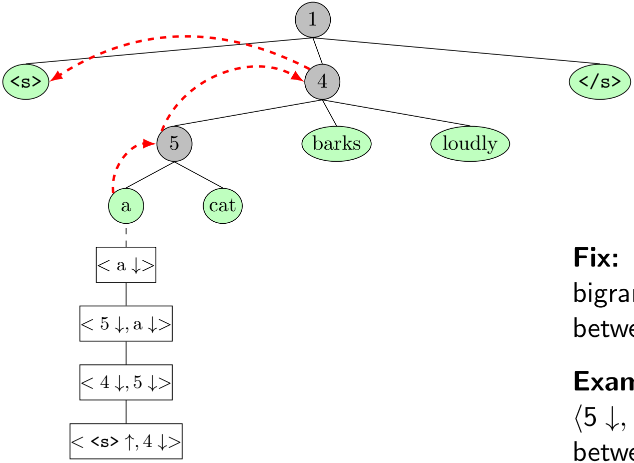
New Constraints: Paths



Fix: In addition to bigrams, consider paths between terminal nodes

Example: Path marker $\langle 5 \downarrow, 10 \downarrow\rangle$ implies that between two word nodes, we move down from node 5 to node 10

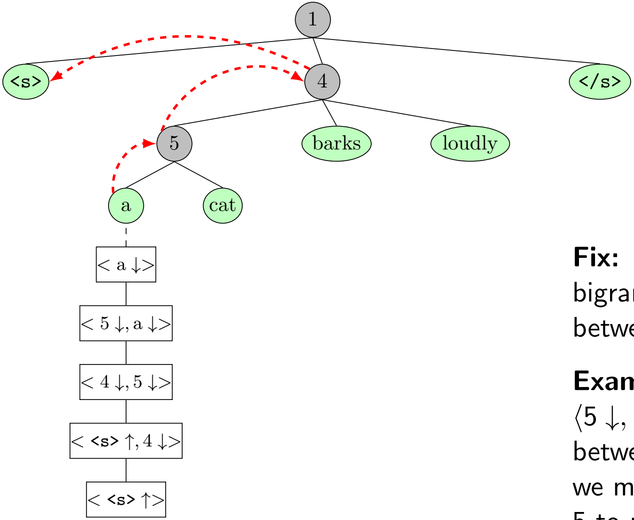
New Constraints: Paths



Fix: In addition to bigrams, consider paths between terminal nodes

Example: Path marker $\langle 5\downarrow, 10\downarrow\rangle$ implies that between two word nodes, we move down from node 5 to node 10

New Constraints: Paths

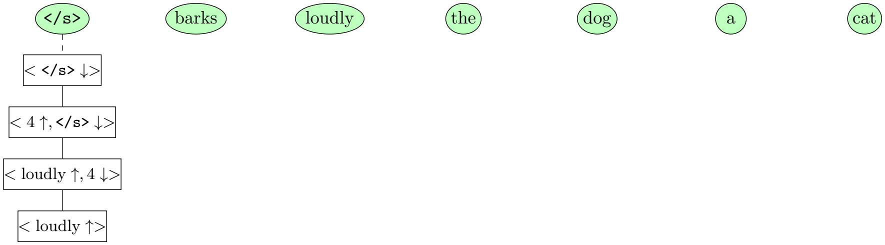


Fix: In addition to bigrams, consider paths between terminal nodes

Example: Path marker $\langle 5 \downarrow, 10 \downarrow\rangle$ implies that between two word nodes, we move down from node 5 to node 10

Greedy Language Model with Paths

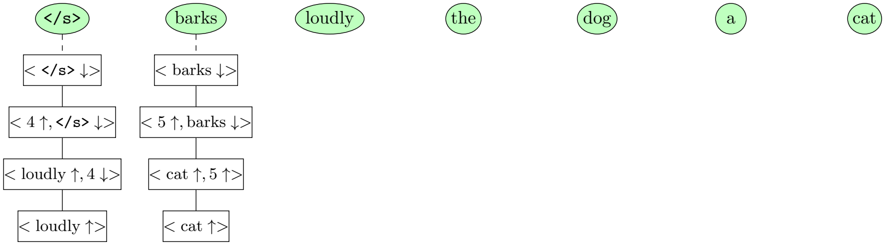
Step 1. Greedily choose best path each word



</s>
barks
loudly
the
dog
a
cat
$< \texttt{</s>} \downarrow >$
$< 4 \uparrow, \texttt{</s>} \downarrow >$
$<$ loudly $\uparrow, 4 \downarrow >$
$<$ loudly $\uparrow >$

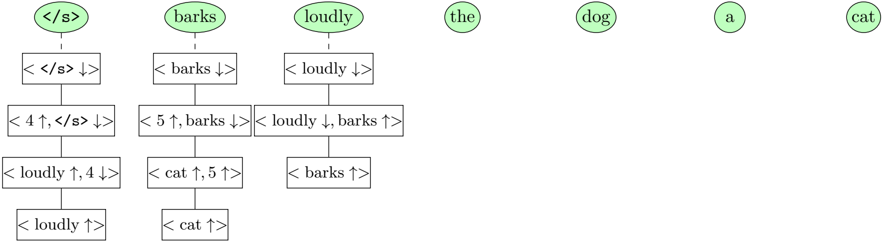
Greedy Language Model with Paths

Step 1. Greedily choose best path each word



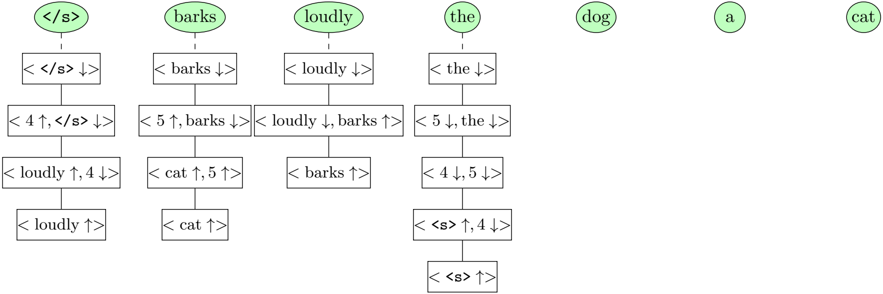
Greedy Language Model with Paths

Step 1. Greedily choose best path each word



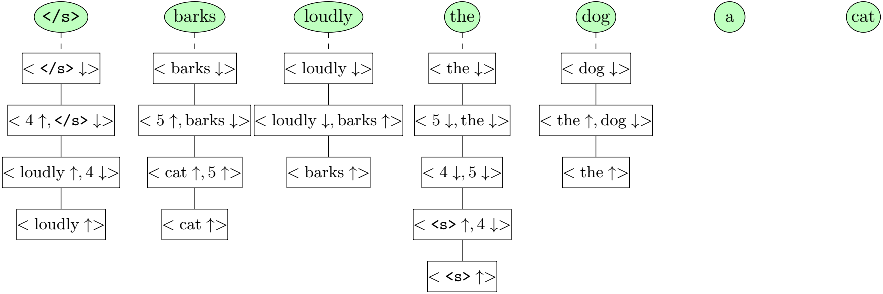
Greedy Language Model with Paths

Step 1. Greedily choose best path each word



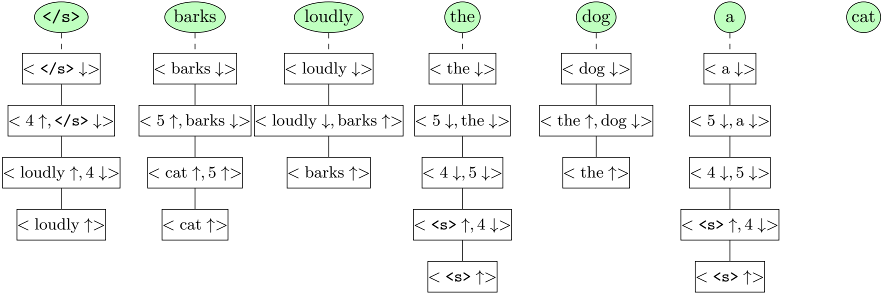
Greedy Language Model with Paths

Step 1. Greedily choose best path each word



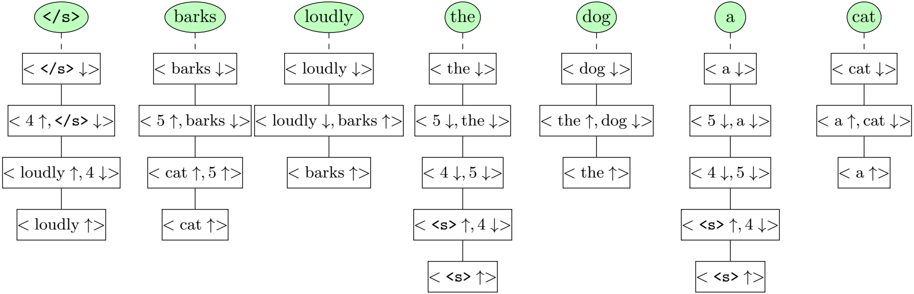
Greedy Language Model with Paths

Step 1. Greedily choose best path each word



Greedy Language Model with Paths

Step 1. Greedily choose best path each word



</s>

< </s> ↓>

< 4 ↑, </s> ↓>

< loudly ↑, 4 ↓>

< loudly ↑>

barks

< barks ↓>

< 5 ↑, barks ↓>

< cat ↑, 5 ↑>

< cat ↑>

loudly

< loudly ↓>

< loudly ↓, barks ↑>

< barks ↑>

the

< the ↓>

< 5 ↓, the ↓>

< 4 ↓, 5 ↓>

< <s> ↑, 4 ↓>

< <s> ↑>

dog

< dog ↓>

< the ↑, dog ↓>

< the ↑>

a

< a ↓>

< 5 ↓, a ↓>

< 4 ↓, 5 ↓>

< <s> ↑, 4 ↓>

< <s> ↑>

cat

< cat ↓>

< a ↑, cat ↓>

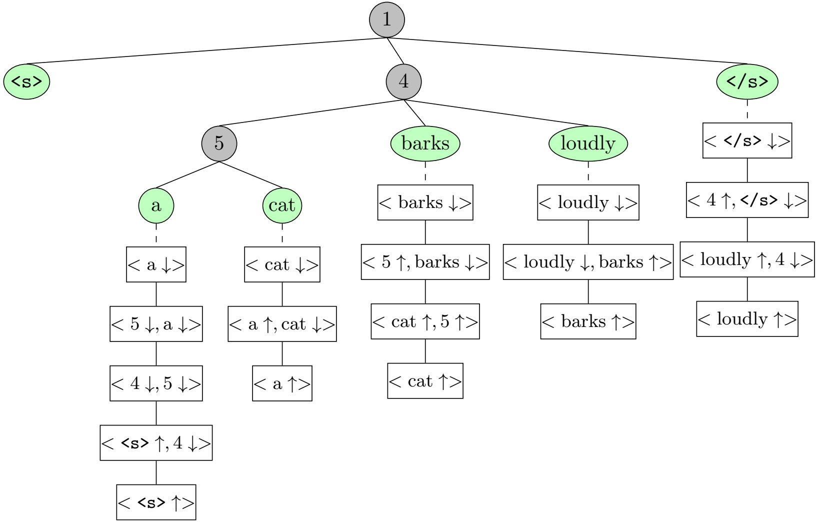
< a ↑>

Greedy Language Model with Paths (continued)

Step 2. Find the best derivation over these elements

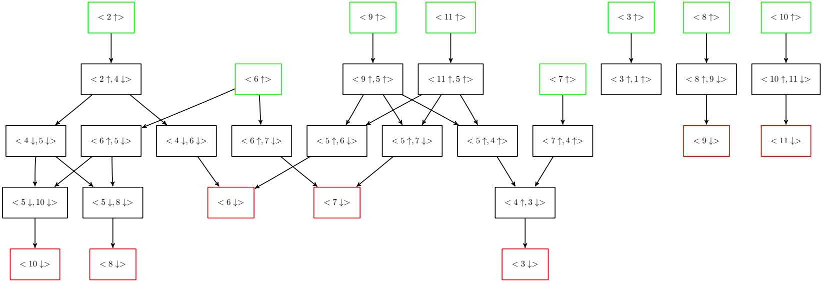
Greedy Language Model with Paths (continued)

Step 2. Find the best derivation over these elements



Efficiently Calculating Best Paths

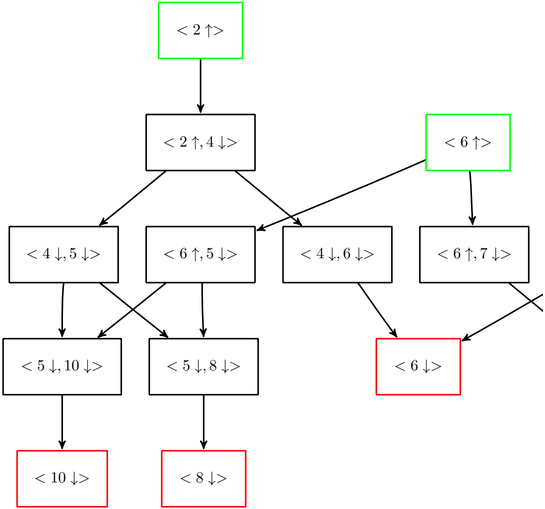
There are too many paths to compute argmax directly, but we can compactly represent all paths as a graph



Graph is linear in the size of the grammar

- Green nodes represent leaving a word
- Red nodes represent entering a word
- Black nodes are intermediate paths

Best Paths


$< 2 \uparrow>$
$< 2 \uparrow, 4 \downarrow>$
$< 6 \uparrow>$
$< 4 \downarrow, 5 \downarrow>$
$< 6 \uparrow, 5 \downarrow>$
$< 4 \downarrow, 6 \downarrow>$
$< 6 \uparrow, 7 \downarrow>$
$< 5 \downarrow, 10 \downarrow>$
$< 5 \downarrow, 8 \downarrow>$
$< 6 \downarrow>$
$< 10 \downarrow>$
$< 8 \downarrow>$

Goal: Find the best path between all word nodes (green and red)

Method: Run all-pairs shortest path to find best paths

Full Algorithm

Algorithm is very similar to simple bigram case. Penalty weights are associated with nodes in the graph instead of just bigram words

Theorem

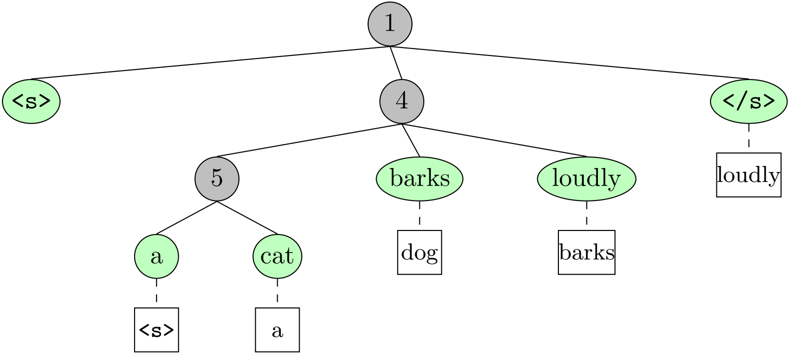
If at any iteration the greedy paths agree with the derivation, then $(y^{(k)})$ is the global optimum.

But what if it does not find the global optimum?

Convergence

The algorithm is not guaranteed to converge

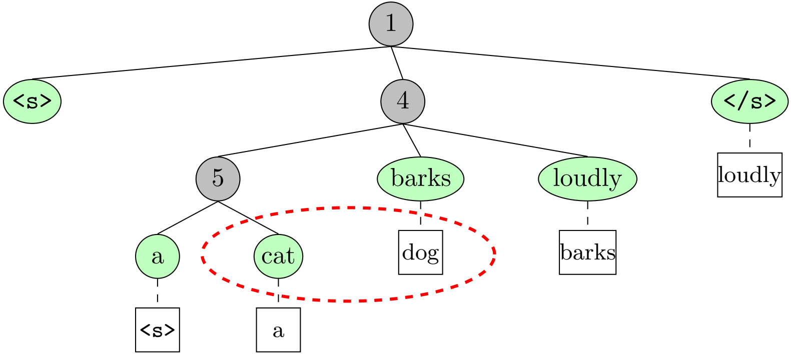
May get stuck between solutions.



Convergence

The algorithm is not guaranteed to converge

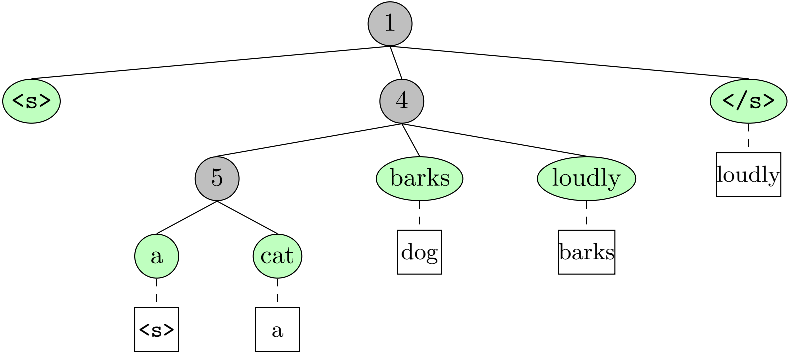
May get stuck between solutions.



Convergence

The algorithm is not guaranteed to converge

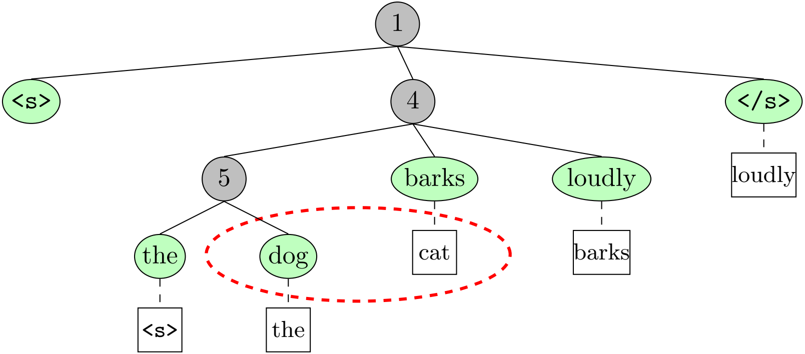
May get stuck between solutions.



Convergence

The algorithm is not guaranteed to converge

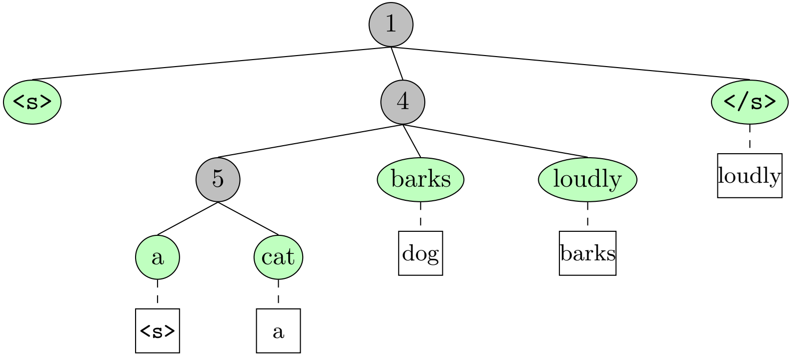
May get stuck between solutions.



Convergence

The algorithm is not guaranteed to converge

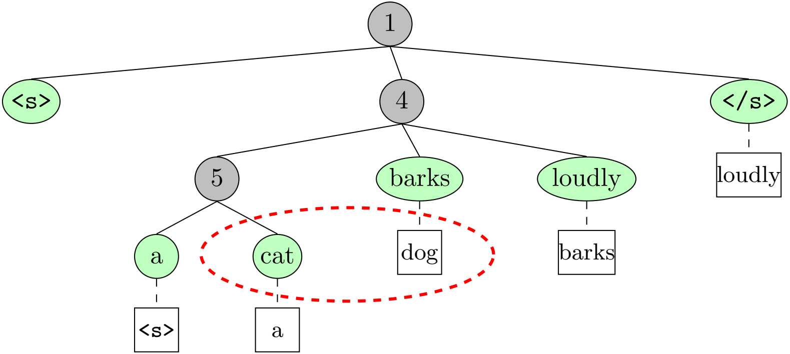
May get stuck between solutions.



Convergence

The algorithm is not guaranteed to converge

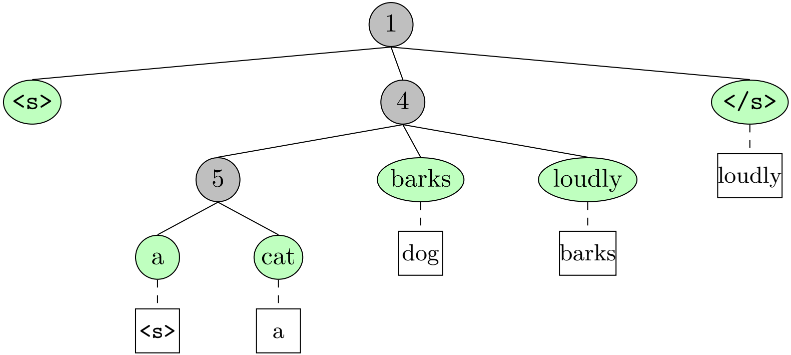
May get stuck between solutions.



Convergence

The algorithm is not guaranteed to converge

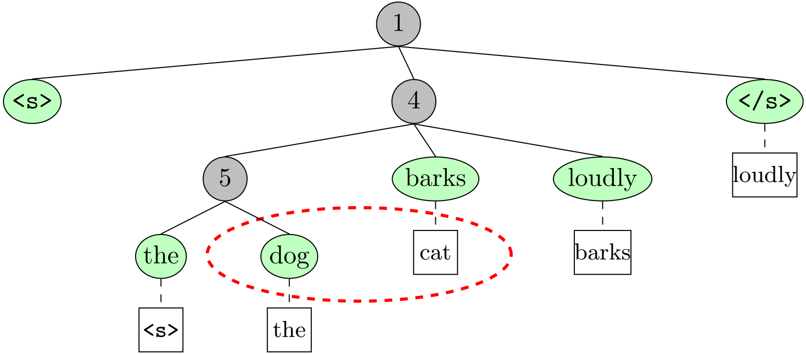
May get stuck between solutions.



Convergence

The algorithm is not guaranteed to converge

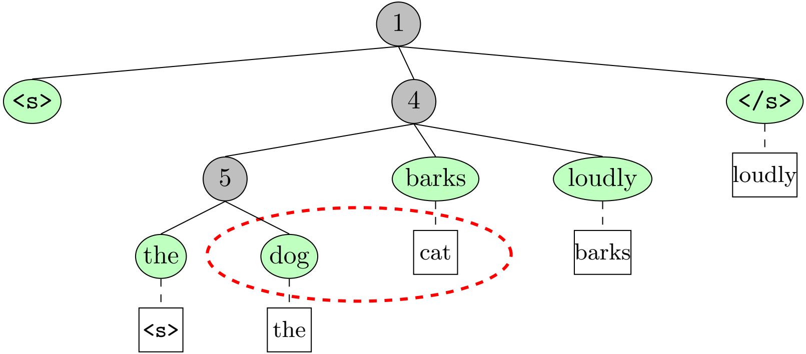
May get stuck between solutions.



Convergence

The algorithm is not guaranteed to converge

May get stuck between solutions.



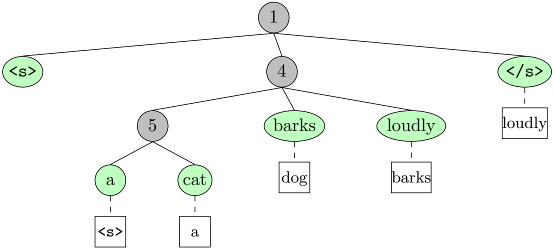
Can fix this by incrementally adding constraints to the problem

Tightening

Main idea: Keep partition sets (A and B). The parser treats all words in a partition as the same word.

- Initially place all words in the same partition.
- If the algorithm gets stuck, separate words that conflict
- Run the exact algorithm but only distinguish between partitions (much faster than running full exact algorithm)

Example:



Partitions

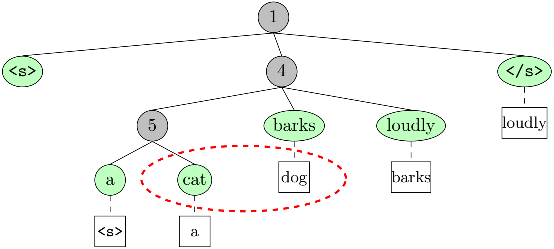
A = {2,6,7,8,9,10,11}
B = {}

Tightening

Main idea: Keep partition sets (A and B). The parser treats all words in a partition as the same word.

- Initially place all words in the same partition.
- If the algorithm gets stuck, separate words that conflict
- Run the exact algorithm but only distinguish between partitions (much faster than running full exact algorithm)

Example:



Partitions

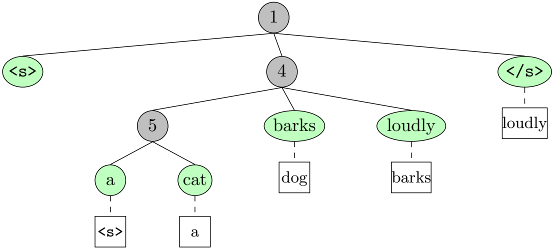
A = {2,6,7,8,9,10,11}
B = {}

Tightening

Main idea: Keep partition sets (A and B). The parser treats all words in a partition as the same word.

- Initially place all words in the same partition.
- If the algorithm gets stuck, separate words that conflict
- Run the exact algorithm but only distinguish between partitions (much faster than running full exact algorithm)

Example:



Partitions

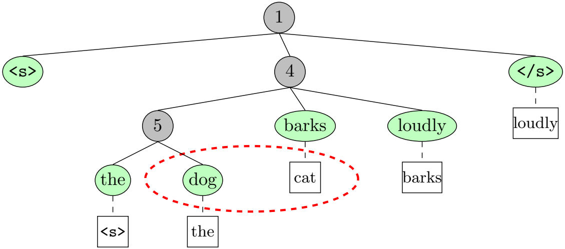
A = {2,6,7,8,9,10,11}
B = {}

Tightening

Main idea: Keep partition sets (A and B). The parser treats all words in a partition as the same word.

- Initially place all words in the same partition.
- If the algorithm gets stuck, separate words that conflict
- Run the exact algorithm but only distinguish between partitions (much faster than running full exact algorithm)

Example:



Partitions

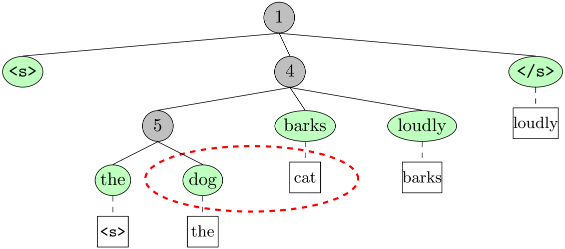
A = {2,6,7,8,9,10,11}
B = {}

Tightening

Main idea: Keep partition sets (A and B). The parser treats all words in a partition as the same word.

- Initially place all words in the same partition.
- If the algorithm gets stuck, separate words that conflict
- Run the exact algorithm but only distinguish between partitions (much faster than running full exact algorithm)

Example:



Partitions

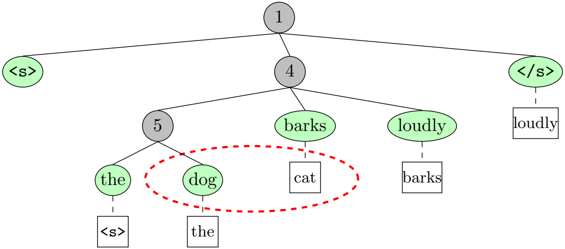
A = {2,6,7,8,9,10}
B = {11}

Tightening

Main idea: Keep partition sets (A and B). The parser treats all words in a partition as the same word.

- Initially place all words in the same partition.
- If the algorithm gets stuck, separate words that conflict
- Run the exact algorithm but only distinguish between partitions (much faster than running full exact algorithm)

Example:



Partitions

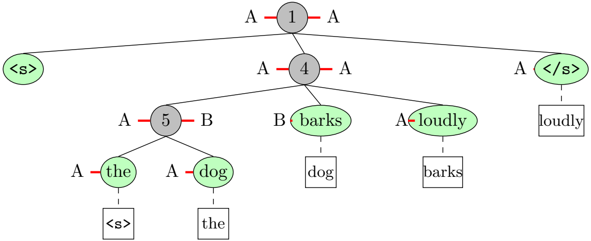
A = {2,6,7,8,9,10}
B = {11}

Tightening

Main idea: Keep partition sets (A and B). The parser treats all words in a partition as the same word.

- Initially place all words in the same partition.
- If the algorithm gets stuck, separate words that conflict
- Run the exact algorithm but only distinguish between partitions (much faster than running full exact algorithm)

Example:



Partitions

A = {2,6,7,8,9,10}

B = {11}

Experiments

Properties:

- Exactness
- Translation Speed
- Comparison to Cube Pruning

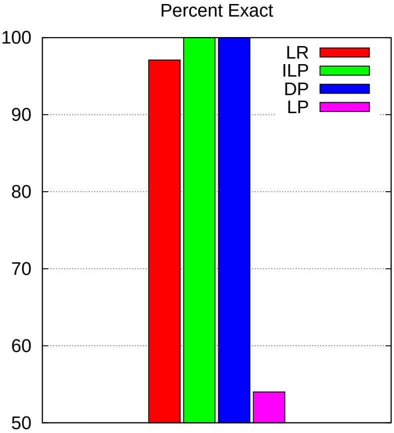
Model:

- Tree-to-String translation model (Huang and Mi, 2010)
- Trained with MERT

Experiments:

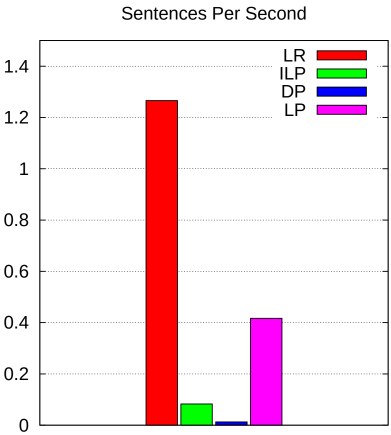
- NIST MT Evaluation Set (2008)

Exactness



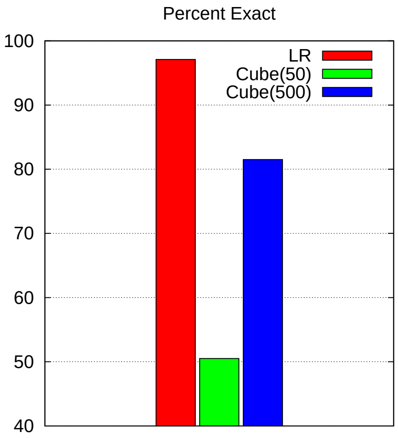
LR	Lagrangian Relaxation
ILP	Integer Linear Programming
DP	Exact Dynanic Programming
LP	Linear Programming

Median Speed



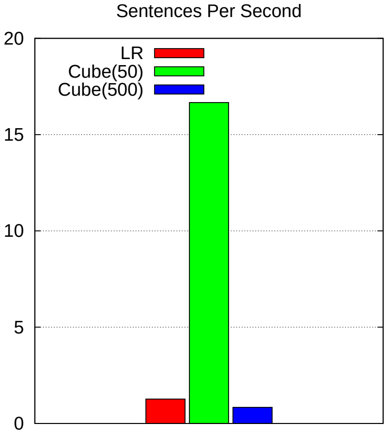
LR	Lagrangian Relaxation
ILP	Integer Linear Programming
DP	Exact Dynanic Programming
LP	Linear Programming

Comparison to Cube Pruning: Exactness



LR	Lagrangian Relaxation
Cube(50)	Cube Pruning (Beam=50)
Cube(500)	Cube Pruning (Beam=500)

Comparison to Cube Pruning: Median Speed



LR	Lagrangian Relaxation
Cube(50)	Cube Pruning (Beam=50)
Cube(500)	Cube Pruning (Beam=500)

The Phrase-Based Decoding Problem

- We have a source-language sentence $x_1, x_2, \ldots, x_N$ (x_i is the i'th word in the sentence)
- A phrase p is a tuple (s, t, e) signifying that words $x_s \ldots x_t$ have a target-language translation as e
- E.g., $p = (2, 5, \textit{the dog})$ specifies that words $x_2 \ldots x_5$ have a translation as *the dog*
- Output from a phrase-based model is a *derivation*

$$y = p_1 p_2 \ldots p_L$$

where p_j for $j = 1 \ldots L$ are phrases. A derivation defines a translation $e(y)$ formed by concatenating the strings

$$e(p_1)e(p_2)\ldots e(p_L)$$

Scoring Derivations

- Each phrase p has a score $g(p)$.

- For two consecutive phrases $p_k = (s, t, e)$ and $p_{k+1} = (s', t', e')$, the *distortion distance* is $\delta(t, s') = |t + 1 - s'|$

- The score for a derivation is

$$f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta \times \delta(t(p_k), s(p_{k+1}))$$

where $\eta \in \mathbb{R}$ is the distortion penalty, and $h(e(y))$ is the language model score

The Decoding Problem

- $\mathcal{Y}$ is the set of all valid derivations
- For a derivation y, $y(i)$ is the number of times word i is translated
- A derivation $y = p_1, p_2, \ldots, p_L$ is valid if:
  - $y(i) = 1$ for $i = 1 \ldots N$
  - For each pair of consecutive phrases p_k, p_{k+1} for $k = 1 \ldots L-1$, we have $\delta(t(p_k), s(p_{k+1})) \leq d$, where d is the *distortion limit*.
- Decoding problem is to find

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

Exact Dynamic Programming

- We can find

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

  using dynamic programming

- **But**, the runtime (and number of states) is exponential in N.

- Dynamic programming states are of the form

  $$(w_1, w_2, b, r)$$

  where

  - w_1, w_2 are last two words of a hypothesis
  - b is a bit-string of length N, recording which words have been translated (2^N possibilities)
  - r is the end-point of the last phrase in the hypothesis

A Lagrangian Relaxation Algorithm

- Define $\mathcal{Y}'$ to be the set of derivations such that:
  - $\sum_{i=1}^{N} y(i) = N$
  - For each pair of consecutive phrases p_k, p_{k+1} for $k = 1 \ldots L-1$, we have $\delta(t(p_k), s(p_{k+1})) \leq d$, where d is the *distortion limit*.

- Notes:
  - We have dropped the $y(i) = 1$ constraints.
  - We have $\mathcal{Y} \subset \mathcal{Y}'$

Dynamic Programming over $\mathcal{Y}'$

- We can find
$$\arg\max_{y \in \mathcal{Y}'} f(y)$$
efficiently, using dynamic programming

- Dynamic programming states are of the form
$$(w_1, w_2, n, r)$$
where
  - w_1, w_2 are last two words of a hypothesis
  - n is the length of the partial hypothesis
  - r is the end-point of the last phrase in the hypothesis

A Lagrangian Relaxation Algorithm (continued)

- The original decoding problem is

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

- We can rewrite this as

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

- We deal with the $y(i) = 1$ constraints using Lagrangian relaxation

A Lagrangian Relaxation Algorithm (continued)

The Lagrangian is

$$L(u, y) = f(y) + \sum_i u(i)(y(i) - 1)$$

The dual objective is then

$$L(u) = \max_{y \in \mathcal{Y}'} L(u, y).$$

and the dual problem is to solve

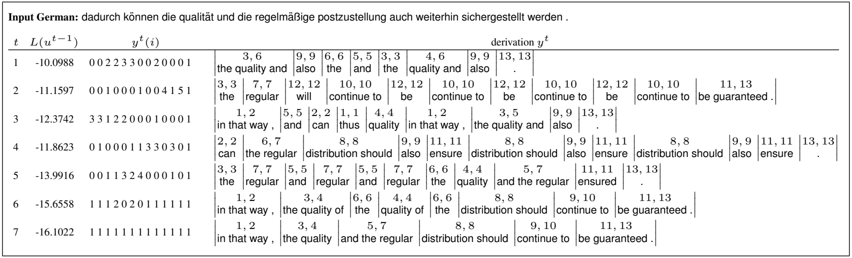
$$\min_u L(u).$$

The Algorithm

```
Initialization: $u^0(i) \leftarrow 0$    for $i = 1 \ldots N$
for $t = 1 \ldots T$
    $y^t = \mathrm{argmax}_{y \in \mathcal{Y}'} L(u^{t-1}, y)$
    if $y^t(i) = 1$  for  $i = 1 \ldots N$
        return $y^t$
    else
        for $i = 1 \ldots N$
            $u^t(i) = u^{t-1}(i) - \alpha^t \left(y^t(i) - 1\right)$
```

Figure: The decoding algorithm. $\alpha^t > 0$ is the step size at the t'th iteration.

An Example Run of the Algorithm



Input German: dadurch können die qualität und die regelmäßige postzustellung auch weiterhin sichergestellt werden .

t	$L(u^{t-1})$	$y^t(i)$	derivation y^t											
1	-10.0988	0 0 2 2 3 3 0 0 2 0 0 0 1	$3, 6$ the quality and	$9, 9$ also	$6, 6$ the	$5, 5$ and	$3, 3$ the	$4, 6$ quality and	$9, 9$ also	$13, 13$.				
2	-11.1597	0 0 1 0 0 0 1 0 0 4 1 5 1	$3, 3$ the	$7, 7$ regular	$12, 12$ will	$10, 10$ continue to	$12, 12$ be	$10, 10$ continue to	$12, 12$ be	$10, 10$ continue to	$12, 12$ be	$10, 10$ continue to	$11, 13$ be guaranteed .	
3	-12.3742	3 3 1 2 2 0 0 0 1 0 0 0 1	$1, 2$ in that way ,	$5, 5$ and	$2, 2$ can	$1, 1$ thus	$4, 4$ quality	$1, 2$ in that way ,	$3, 5$ the quality and	$9, 9$ also	$13, 13$.			
4	-11.8623	0 1 0 0 0 1 1 3 3 0 3 0 1	$2, 2$ can	$6, 7$ the regular	$8, 8$ distribution should	$9, 9$ also	$11, 11$ ensure	$8, 8$ distribution should	$9, 9$ also	$11, 11$ ensure	$8, 8$ distribution should	$9, 9$ also	$11, 11$ ensure	$13, 13$.
5	-13.9916	0 0 1 1 3 2 4 0 0 0 1 0 1	$3, 3$ the	$7, 7$ regular	$5, 5$ and	$7, 7$ regular	$5, 5$ and	$7, 7$ regular	$6, 6$ the	$4, 4$ quality	$5, 7$ and the regular	$11, 11$ ensured	$13, 13$.	
6	-15.6558	1 1 1 2 0 2 0 1 1 1 1 1 1	$1, 2$ in that way ,	$3, 4$ the quality of	$6, 6$ the	$4, 4$ quality of	$6, 6$ the	$8, 8$ distribution should	$9, 10$ continue to	$11, 13$ be guaranteed .				
7	-16.1022	1 1 1 1 1 1 1 1 1 1 1 1 1	$1, 2$ in that way ,	$3, 4$ the quality	$5, 7$ and the regular	$8, 8$ distribution should	$9, 10$ continue to	$11, 13$ be guaranteed .						

Tightening the Relaxation

- In some cases, the relaxation is not tight, and the algorithm will not converge to $y(i) = 1$ for $i = 1 \ldots N$
- Our solution: incrementally add *hard constraints* until the relaxation is tight
- Definition: for any set $\mathcal{C} \subseteq \{1, 2, \ldots, N\}$,

$$\mathcal{Y}'_{\mathcal{C}} = \{y : y \in \mathcal{Y}', \text{ and } \forall i \in \mathcal{C}, y(i) = 1\}$$

- We can find

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

  using dynamic programming, with a $2^{|\mathcal{C}|}$ increase in the number of states

- Goal: find a small set $\mathcal{C}$ such that Lagrangian relaxation with $\mathcal{Y}'_{\mathcal{C}}$ returns an exact solution

An Example Run of the Algorithm

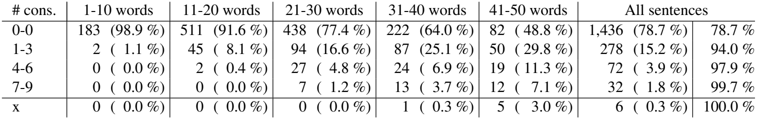
Input German: es bleibt jedoch dabei , dass kolumbien ein land ist , das aufmerksam beobachtet werden muss .

t	$L(u^{t-1})$	$y^t(i)$	derivation y^t												
1	-11.8658	0 0 0 0 1 3 0 3 3 4 1 1 0 0 0 0 1	5, 6 that	10, 10 is	8, 9 a country	6, 6 that	10, 10 is	8, 9 a country	6, 6 that	10, 10 is	8, 8 a	9, 12 country that	17, 17 .		
2	-5.46647	2 2 4 0 2 0 1 0 0 0 1 0 1 1 1 1 1	3, 3 however ,	1, 1 it	2, 3 is , however	5, 5 ,	3, 3 however ,	1, 1 it	2, 3 is , however	5, 5 ,	7, 7 colombia	11, 11 ,	16, 16 must	13, 15 be closely monitored	17, 17 .
	⋮														
32	-17.0203	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	1, 5 nonetheless ,	7, 7 colombia	10, 10 is	8, 8 a	9, 12 country that	16, 16 must	13, 15 be closely monitored	17, 17 .					
33	-17.1727	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	1, 5 nonetheless ,	6, 6 that	8, 9 a country	6, 6 that	7, 7 colombia	11, 12 , which	16, 16 must	13, 15 be closely monitored	17, 17 .				
34	-17.0203	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	1, 5 nonetheless ,	7, 7 colombia	10, 10 is	8, 8 a	9, 12 country that	16, 16 must	13, 15 be closely monitored	17, 17 .					
35	-17.1631	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	1, 5 nonetheless ,	7, 7 colombia	10, 10 is	8, 8 a	9, 12 country that	16, 16 must	13, 15 be closely monitored	17, 17 .					
36	-17.0408	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	1, 5 nonetheless ,	6, 6 that	8, 9 a country	6, 6 that	7, 7 colombia	11, 12 , which	16, 16 must	13, 15 be closely monitored	17, 17 .				
37	-17.1727	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	1, 5 nonetheless ,	7, 7 colombia	10, 10 is	8, 8 a	9, 12 country that	16, 16 must	13, 15 be closely monitored	17, 17 .					
38	-17.0408	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	1, 5 nonetheless ,	6, 6 that	8, 9 a country	6, 6 that	7, 7 colombia	11, 12 , which	16, 16 must	13, 15 be closely monitored	17, 17 .				
39	-17.1658	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	1, 5 nonetheless ,	6, 6 that	8, 9 a country	6, 6 that	7, 7 colombia	11, 12 , which	16, 16 must	13, 15 be closely monitored	17, 17 .				
40	-17.056	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	1, 5 nonetheless ,	7, 7 colombia	10, 10 is	8, 8 a	9, 12 country that	16, 16 must	13, 15 be closely monitored	17, 17 .					
41	-17.1732	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	1, 5 nonetheless ,	6, 6 that	8, 9 a country	6, 6 that	7, 7 colombia	11, 12 , which	16, 16 must	13, 15 be closely monitored	17, 17 .				
		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	$count(6) = 10; count(10) = 10; count(i) = 0$ for all other i **adding constraints: 6 10**												
42	-17.229	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1, 5 nonetheless ,	7, 7 colombia	6, 6 that	8, 12 a country that	16, 16 must	13, 15 be closely monitored	17, 17 .						

The Algorithm with Constraint Generation

```
$Optimize(\mathcal{C}, u)$
  while (dual value still improving)
    $y^* = \text{argmax}_{y \in \mathcal{Y}'_{\mathcal{C}}} L(u, y)$
    if $y^*(i) = 1$ for $i = 1 \ldots N$    return $y^*$
    else for $i = 1 \ldots N$
        $u(i) = u(i) - \alpha\,(y^*(i) - 1)$
  $count(i) = 0$ for $i = 1 \ldots N$
  for $k = 1 \ldots K$
    $y^* = \text{argmax}_{y \in \mathcal{Y}'_{\mathcal{C}}} L(u, y)$
    if $y^*(i) = 1$ for $i = 1 \ldots N$    return $y^*$
    else for $i = 1 \ldots N$
        $u(i) = u(i) - \alpha\,(y^*(i) - 1)$
        $count(i) = count(i) + [[y^*(i) \neq 1]]$
  Let $\mathcal{C}'$ = set of G $i$'s that have the largest value for $count(i)$ and that are not in $\mathcal{C}$
  return $Optimize(\mathcal{C} \cup \mathcal{C}', u)$
```

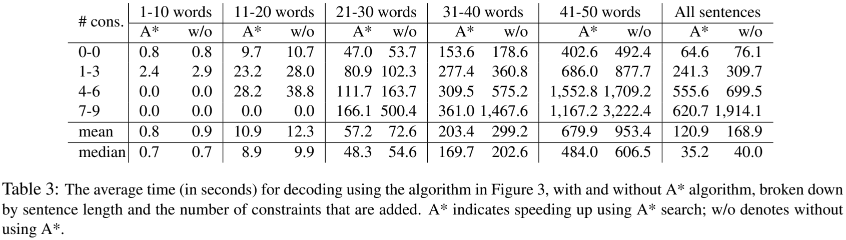
Number of Constraints Required



# cons.	1-10 words	11-20 words	21-30 words	31-40 words	41-50 words	All sentences	
0-0	183 (98.9 %)	511 (91.6 %)	438 (77.4 %)	222 (64.0 %)	82 (48.8 %)	1,436 (78.7 %)	78.7 %
1-3	2 (1.1 %)	45 (8.1 %)	94 (16.6 %)	87 (25.1 %)	50 (29.8 %)	278 (15.2 %)	94.0 %
4-6	0 (0.0 %)	2 (0.4 %)	27 (4.8 %)	24 (6.9 %)	19 (11.3 %)	72 (3.9 %)	97.9 %
7-9	0 (0.0 %)	0 (0.0 %)	7 (1.2 %)	13 (3.7 %)	12 (7.1 %)	32 (1.8 %)	99.7 %
x	0 (0.0 %)	0 (0.0 %)	0 (0.0 %)	1 (0.3 %)	5 (3.0 %)	6 (0.3 %)	100.0 %

Table 2: Table showing the number of constraints added before convergence of the algorithm in Figure 3, broken down by sentence length. Note that a maximum of 3 constraints are added at each recursive call, but that fewer than 3 constraints are added in cases where fewer than 3 constraints have $count(i) > 0$. x indicates the sentences that fail to converge after 250 iterations. 78.7% of the examples converge without adding any constraints.

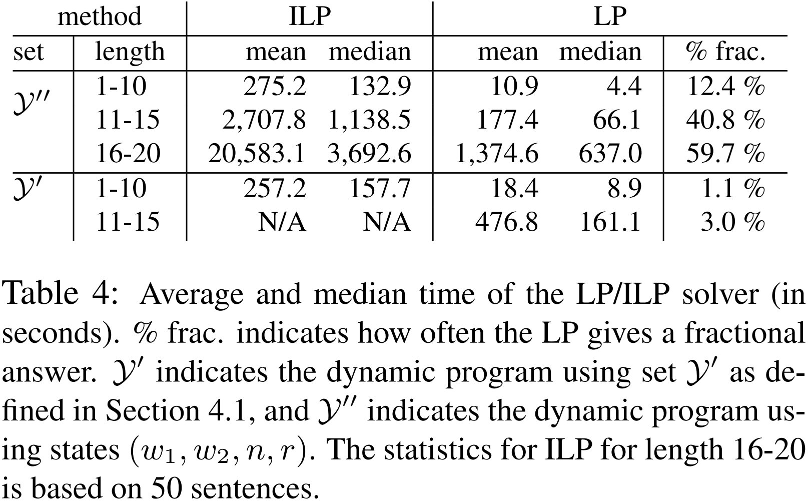
Time Required



# cons.	1-10 words		11-20 words		21-30 words		31-40 words		41-50 words		All sentences	
	A*	w/o	A*	w/o	A*	w/o	A*	w/o	A*	w/o	A*	w/o
0-0	0.8	0.8	9.7	10.7	47.0	53.7	153.6	178.6	402.6	492.4	64.6	76.1
1-3	2.4	2.9	23.2	28.0	80.9	102.3	277.4	360.8	686.0	877.7	241.3	309.7
4-6	0.0	0.0	28.2	38.8	111.7	163.7	309.5	575.2	1,552.8	1,709.2	555.6	699.5
7-9	0.0	0.0	0.0	0.0	166.1	500.4	361.0	1,467.6	1,167.2	3,222.4	620.7	1,914.1
mean	0.8	0.9	10.9	12.3	57.2	72.6	203.4	299.2	679.9	953.4	120.9	168.9
median	0.7	0.7	8.9	9.9	48.3	54.6	169.7	202.6	484.0	606.5	35.2	40.0

Table 3: The average time (in seconds) for decoding using the algorithm in Figure 3, with and without A* algorithm, broken down by sentence length and the number of constraints that are added. A* indicates speeding up using A* search; w/o denotes without using A*.

Comparison to LP/ILP Decoding



method		ILP		LP		
set	length	mean	median	mean	median	% frac.
$\mathcal{Y}''$	1-10	275.2	132.9	10.9	4.4	12.4 %
	11-15	2,707.8	1,138.5	177.4	66.1	40.8 %
	16-20	20,583.1	3,692.6	1,374.6	637.0	59.7 %
$\mathcal{Y}'$	1-10	257.2	157.7	18.4	8.9	1.1 %
	11-15	N/A	N/A	476.8	161.1	3.0 %

Table 4: Average and median time of the LP/ILP solver (in seconds). % frac. indicates how often the LP gives a fractional answer. $\mathcal{Y}'$ indicates the dynamic program using set $\mathcal{Y}'$ as defined in Section 4.1, and $\mathcal{Y}''$ indicates the dynamic program using states (w_1, w_2, n, r). The statistics for ILP for length 16-20 is based on 50 sentences.

Number of Iterations Required

# iter.	1-10 words	11-20 words	21-30 words	31-40 words	41-50 words	All sentences	
0-7	166 (89.7 %)	219 (39.2 %)	34 (6.0 %)	2 (0.6 %)	0 (0.0 %)	421 (23.1 %)	23.1 %
8-15	17 (9.2 %)	187 (33.5 %)	161 (28.4 %)	30 (8.6 %)	3 (1.8 %)	398 (21.8 %)	44.9 %
16-30	1 (0.5 %)	93 (16.7 %)	208 (36.7 %)	112 (32.3 %)	22 (13.1 %)	436 (23.9 %)	68.8 %
31-60	1 (0.5 %)	52 (9.3 %)	105 (18.6 %)	99 (28.5 %)	62 (36.9 %)	319 (17.5 %)	86.3 %
61-120	0 (0.0 %)	7 (1.3 %)	54 (9.5 %)	89 (25.6 %)	45 (26.8 %)	195 (10.7 %)	97.0 %
121-250	0 (0.0 %)	0 (0.0 %)	4 (0.7 %)	14 (4.0 %)	31 (18.5 %)	49 (2.7 %)	99.7 %
x	0 (0.0 %)	0 (0.0 %)	0 (0.0 %)	1 (0.3 %)	5 (3.0 %)	6 (0.3 %)	100.0 %

Table 1: Table showing the number of iterations taken for the algorithm to converge. x indicates sentences that fail to converge after 250 iterations. of the examples converge within 120 iterations.

Summary

presented dual decomposition as a method for decoding in NLP

formal guarantees

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

efficient algorithms

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

widely applicable

- demonstrated algorithms for a wide range of NLP tasks (parsing, tagging, alignment, mt decoding)

References I

Y. Chang and M. Collins. Exact Decoding of Phrase-based Translation Models through Lagrangian Relaxation. In *To appear proc. of EMNLP*, 2011.

J. DeNero and K. Macherey. Model-Based Aligner Combination Using Dual Decomposition. In *Proc. ACL*, 2011.

Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical Programming*, 1:6–25, 1971. ISSN 0025-5610. URL `http://dx.doi.org/10.1007/BF01584070`. 10.1007/BF01584070.

D. Klein and C.D. Manning. Factored A* Search for Models over Sequences and Trees. In *Proc IJCAI*, volume 18, pages 1246–1251. Citeseer, 2003.

N. Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. ISSN 0162-8828.

References II

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. Dual decomposition for parsing with non-projective head automata. In *EMNLP*, 2010. URL `http://www.aclweb.org/anthology/D10-1125`.

B.H. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Verlag, 2008.

C. Lemaréchal. Lagrangian Relaxation. In *Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions [based on a Spring School]*, pages 112–156, London, UK, 2001. Springer-Verlag. ISBN 3-540-42877-1.

Angelia Nedić and Asuman Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19(4):1757–1780, 2009.

Christopher Raphael. Coarse-to-fine dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23: 1379–1390, 2001.

References III

A.M. Rush and M. Collins. Exact Decoding of Syntactic Translation Models through Lagrangian Relaxation. In *Proc. ACL*, 2011.

A.M. Rush, D. Sontag, M. Collins, and T. Jaakkola. On Dual Decomposition and Linear Programming Relaxations for Natural Language Processing. In *Proc. EMNLP*, 2010.

Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero–one programming problems. *Discrete Applied Mathematics*, 52(1):83 – 106, 1994.

D.A. Smith and J. Eisner. Dependency Parsing by Belief Propagation. In *Proc. EMNLP*, pages 145–156, 2008. URL `http://www.aclweb.org/anthology/D08-1016`.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Proc. UAI*, 2008.