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
- 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

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:

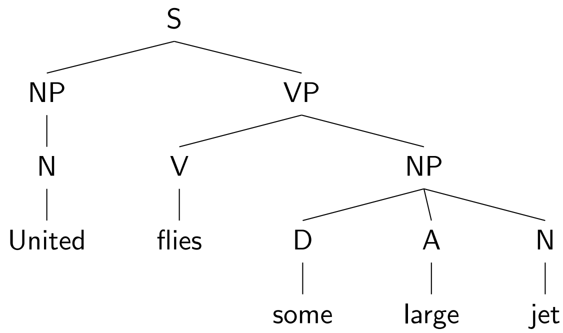
1. worked algorithm for combined parsing and tagging
2. important theorems and formal derivation
3. more examples from parsing, sequence labeling, MT
4. practical considerations for implementing dual decomposition
5. relationship to linear programming relaxations
6. further variations and advanced examples

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

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

$$\ldots + \textit{score}(\mathrm{United}_1, \mathrm{N}) + \textit{score}(\mathrm{V}, \mathrm{N}) + \ldots$$

Constituency parsing

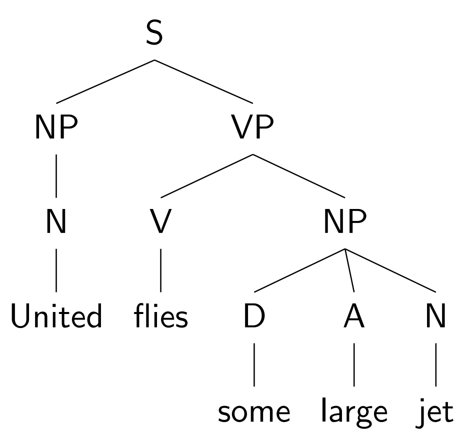
notation:

- $\mathcal{Y}$ 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

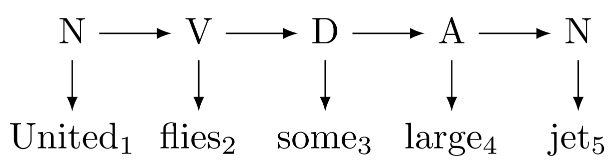
notation:

- $\mathcal{Z}$ 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

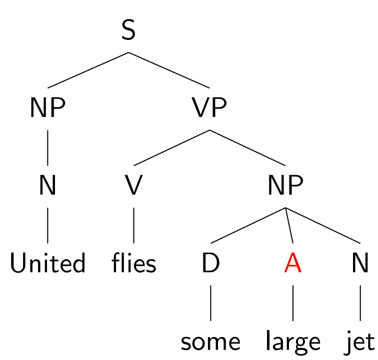


Identifying tags

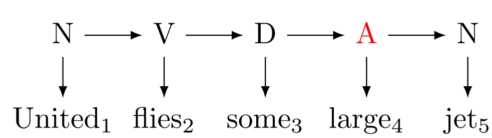
notation: identify the tag labels selected by each model

- $y(i,t) = 1$ 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

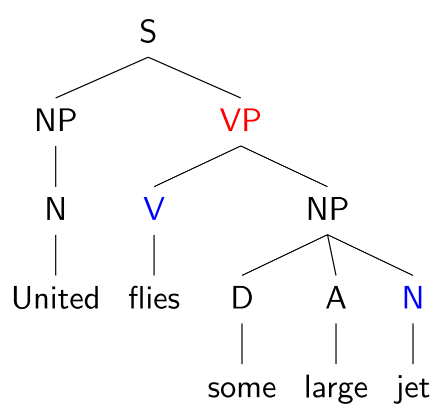
Dynamic programming intersection

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 $S \rightarrow NP\ VP$ with
$S_{N,N} \rightarrow NP_{N,V}\ VP_{V,N}$



Parsing assumption

the structure of $\mathcal{Y}$ is open (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

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

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

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

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

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

Dual decomposition algorithm

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

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

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

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

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

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

Algorithm step-by-step

[Animation]

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

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

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

Lagrangian dual

Lagrangian:

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

Lagrangian dual:

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

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

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

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

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

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

Theorem 2. Convergence

notation:

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

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

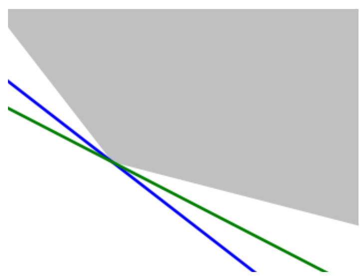
properties:

- 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

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

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

subgradient:

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

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

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

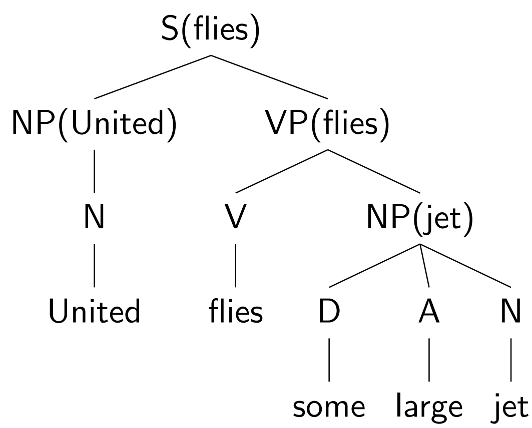
notation:

- $\mathcal{Y}$ 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

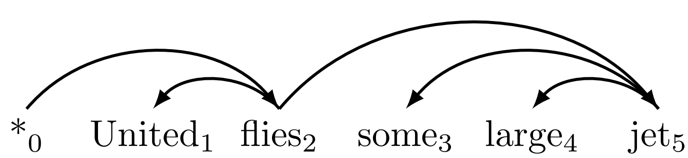


Dependency parsing

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:

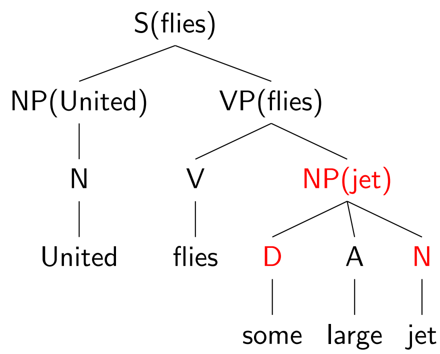


Identifying dependencies

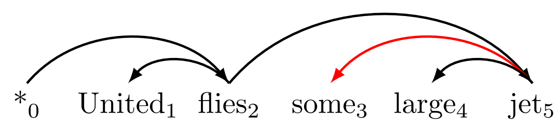
notation: identify the dependencies selected by each model

- $y(i,j) = 1$ when constituency parse y selects word i as a modifier of word j
- $z(i,j) = 1$ when dependency parse z selects word i as a modifier of word j

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 $i = 1 \ldots n$, $j = 0 \ldots n$,

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

Algorithm step-by-step

[Animation]

Corpus-level tagging

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

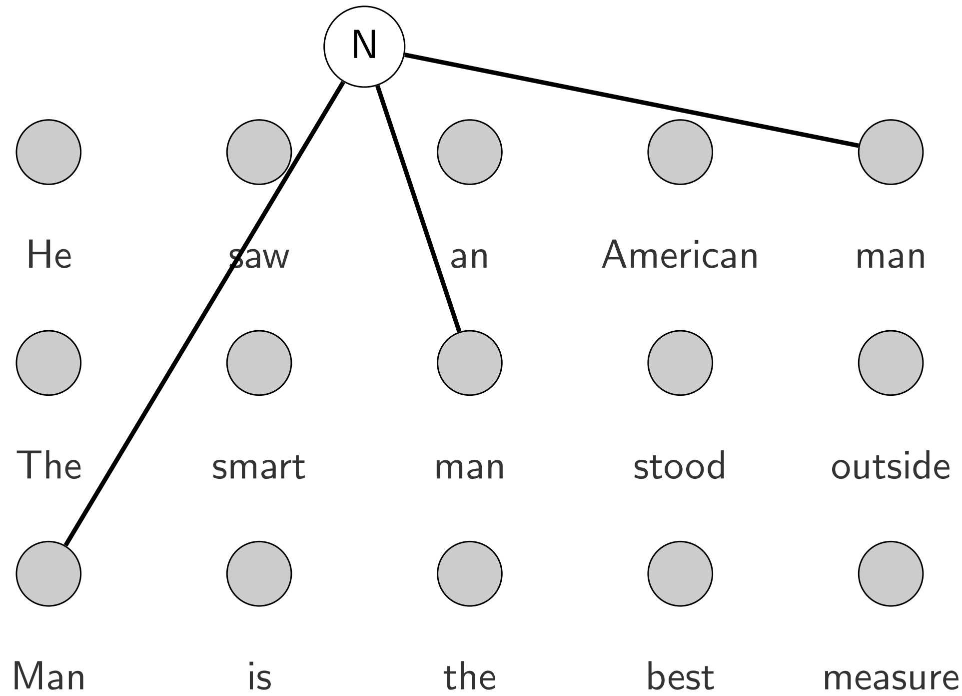
problem: find best tagging for each sentence, while at the same time enforcing inter-sentence soft constraints

example:

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

Corpus-level tagging

full model for corpus-level tagging



Sentence-level decoding

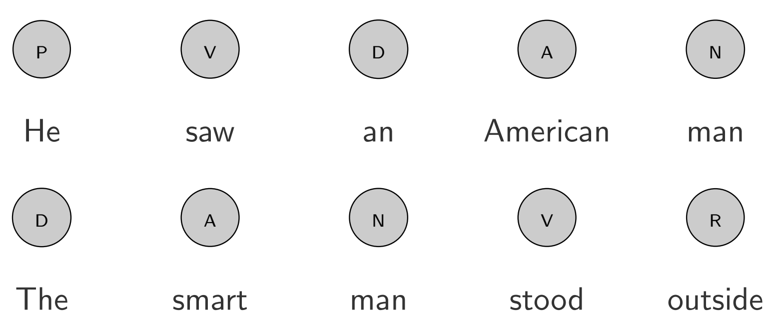
notation:

- $\mathcal{Y}_i$ 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

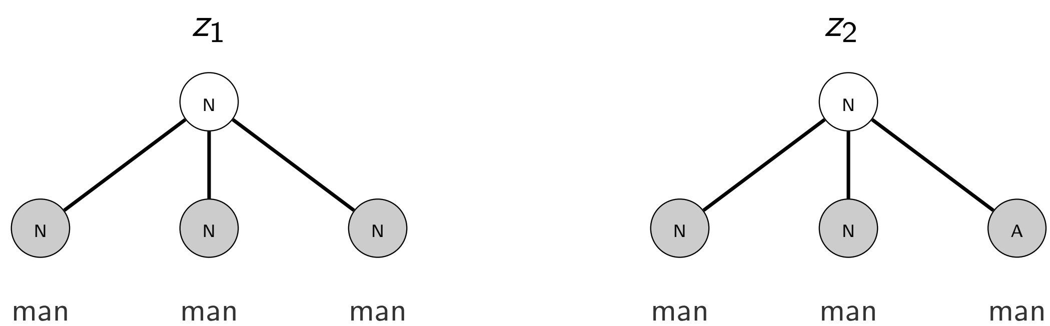


Inter-sentence constraints

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 (e.g. a hard constraint - all word types only have one tag)

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



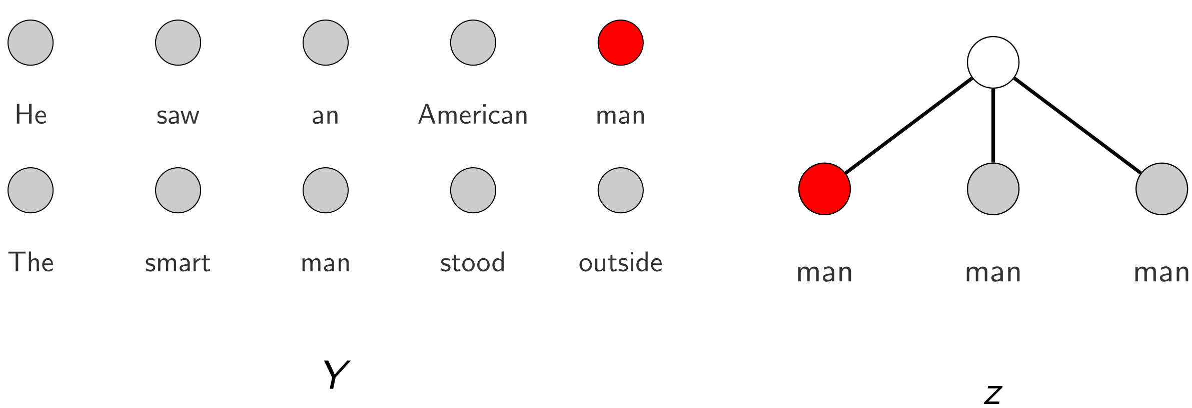
$$g(z_1) > g(z_2)$$

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 $z(1, 5, N) = 1$



Combined optimization

goal:

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

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

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

Algorithm step-by-step

[Animation]

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 with soft agreement

example:

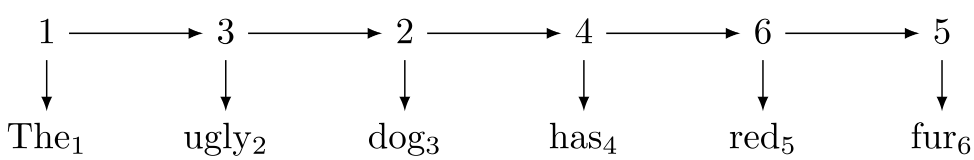
- HMM models for both directional alignments (assume correct alignment is one-to-one for simplicity)

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

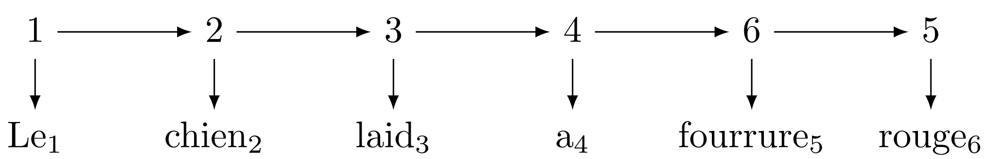


French-to-English alignment

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

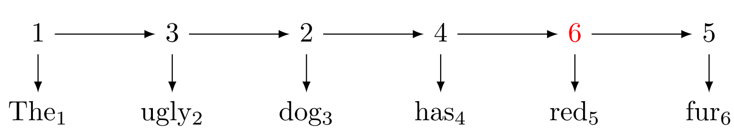


Identifying word alignments

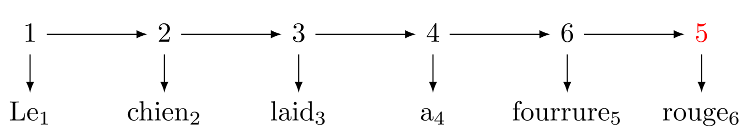
notation: identify the tag labels selected by each model

- $y(i, j) = 1$ 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 $z(6, 5) = 1$



y



z

Combined optimization

goal:

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

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

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

Algorithm step-by-step

[Animation]

4. Practical issues

aim: overview of practical dual decomposition techniques

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

Tracking progress

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

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

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

$$(\min_t L(u^t)) - (f(y^{(k')}) + g(l(y^{(k')})))$$

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

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:

- very helpful if y or z factors naturally into several parts
- decompositions with this property are very fast in practice

example:

- in corpus-level tagging, only need to recompute sentences with a word type that received an update

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:

- Δ_y is the simplex over $\mathcal{Y}$ where $\alpha \in \Delta_y$ implies

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

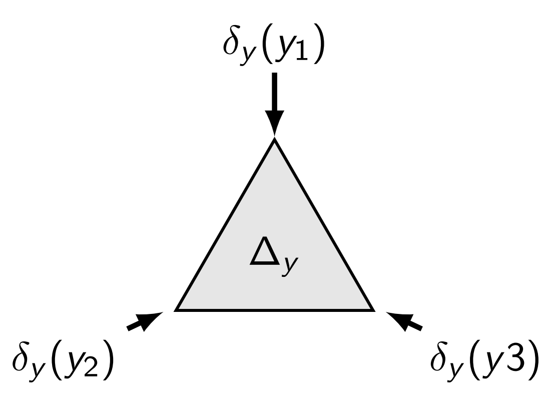
- Δ_z is the simplex over $\mathcal{Z}$
- $\delta_y : \mathcal{Y} \rightarrow \Delta_y$ maps elements to the simplex

example:

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

vertices

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



Linear programming

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

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

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:

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

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

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

Primal relationship

define:

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

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

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

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

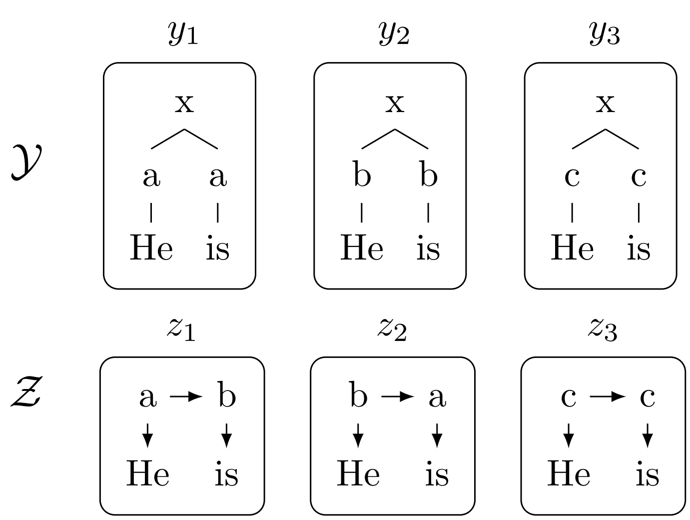
- $\mathcal{Q} \subseteq \mathcal{Q}'$

solutions:

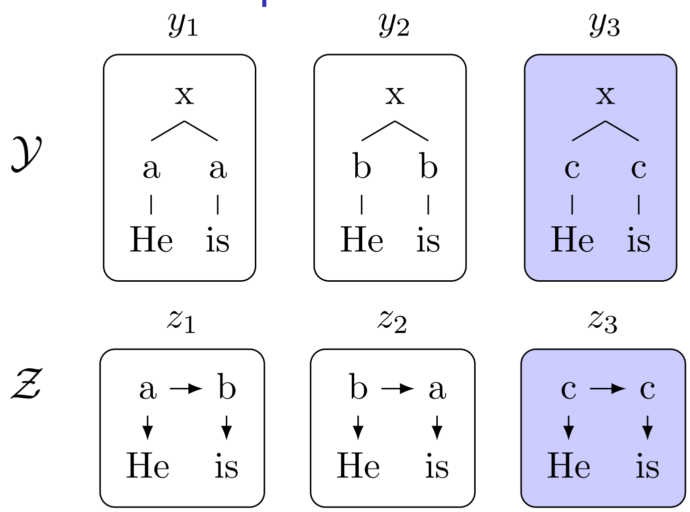
$$\max_{q \in \mathcal{Q}} h(q) \leq \max_{q \in \mathcal{Q}'} h(q) \text{ for any } h$$

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:

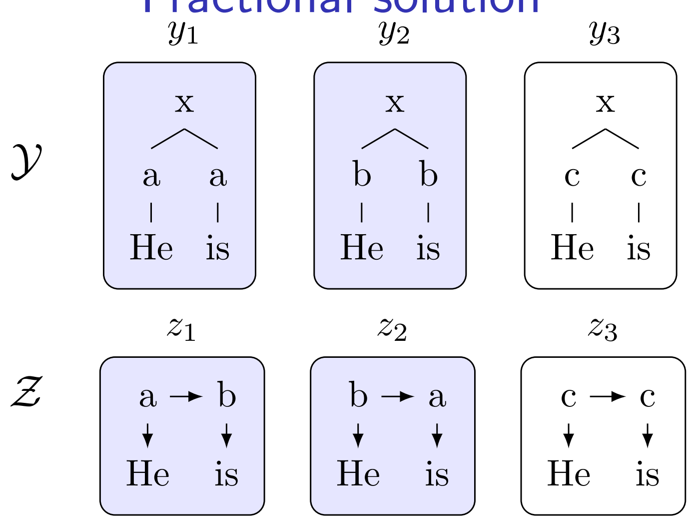
- $\alpha^{(1)} = (0, 0, 1) \in \Delta_y$ is representation of y_3
- $\beta^{(1)} = (0, 0, 1) \in \Delta_z$ is representation of z_3

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

Algorithm run

[Animation]

Tightening (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

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)

Tightening step-by-step

[Animation]

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

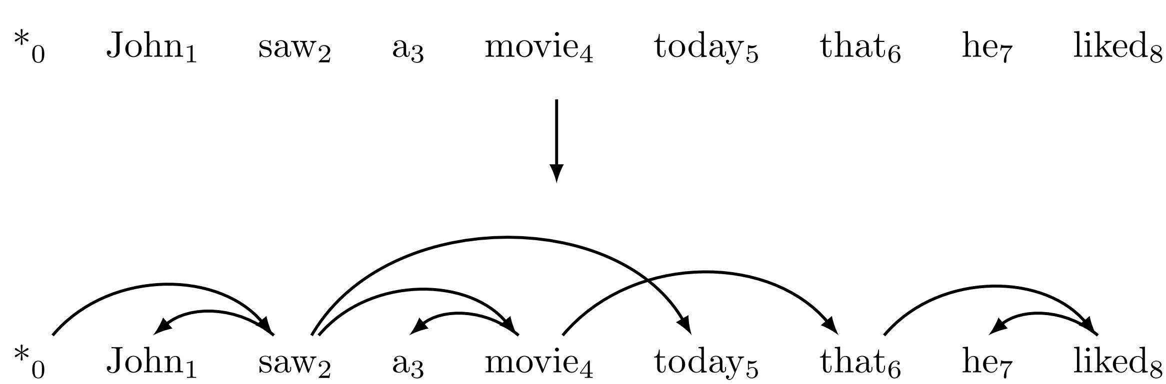
strategy: design a decomposition that separates combinatorial constraints from direct implementation of the scoring function

Non-projective dependency parsing

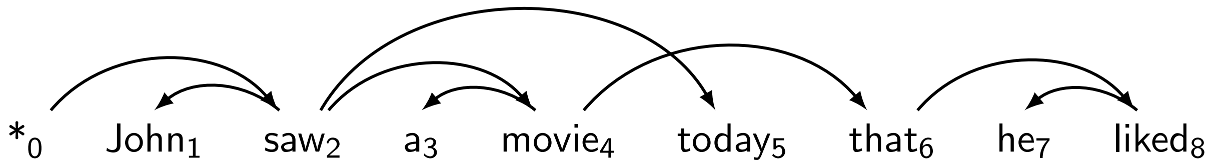
structure:

- starts at the root symbol *
- each word has a exactly one parent word
- produces a tree structure (no cycles)
- dependencies can cross

example:



Arc-Factored



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

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

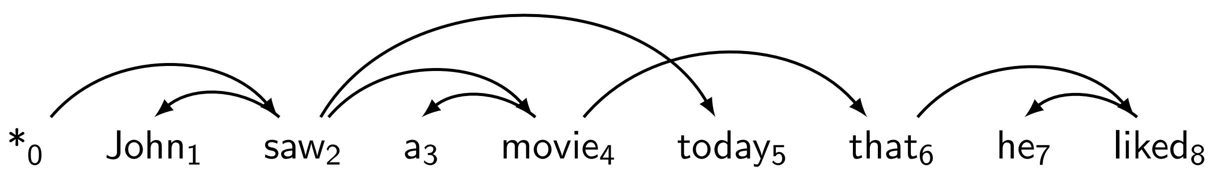
$$+\textit{score}(\text{movie}_4, \text{a}_3) + \ldots$$

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

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

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

Sibling models



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

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

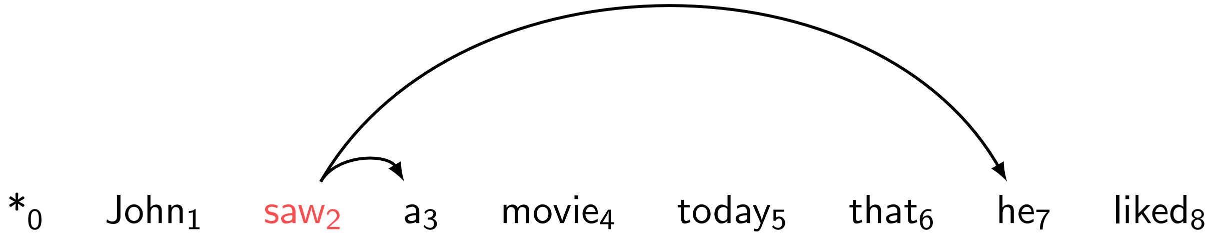
$$+\textit{score}(\text{saw}_2, \text{movie}_4, \text{today}_5) + \ldots$$

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

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

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

Thought experiment: individual decoding



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

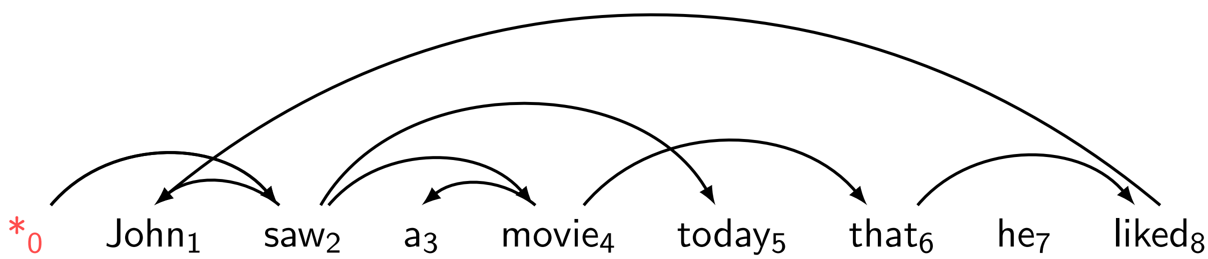
$$\textit{score}(\text{saw}_2, \text{NULL}, \text{John}_1) + \textit{score}(\text{saw}_2, \text{NULL}, \text{movie}_4) + \textit{score}(\text{saw}_2, \text{movie}_4, \text{today}_5)$$

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

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

under sibling model, can solve for each word with Viterbi decoding.

Thought experiment continued



idea: do individual decoding for each head word using dynamic programming

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

but we might violate some constraints

Dual decomposition structure

goal:

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

rewrite:

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

such that for all i, j

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

Algorithm step-by-step

[Animation]

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 example

[Animation]

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

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

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.

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.

References II

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.

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.

References III

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.