Incremental Integer Linear Programming for Non-projective Dependency Parsing

Sebastian Riedel James Clarke

ICCS, University of Edinburgh

22. July 2006
EMNLP 2006

Labelled Dependency Parsing

Labelled Dependency Parsing

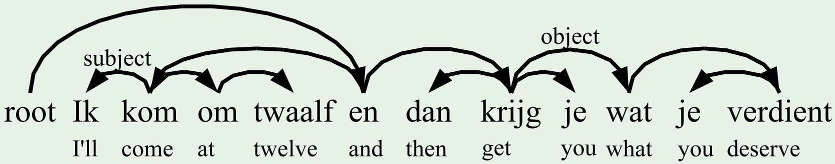
Find labelled head-child relations between tokens.

Labelled Dependency Parsing

Labelled Dependency Parsing

Find labelled head-child relations between tokens.

Example



Non-projective Dependency Parsing

Non-projective Dependency Parsing

Dependencies are allowed to cross

Non-projective Dependency Parsing

Non-projective Dependency Parsing

Dependencies are allowed to cross

Example

Non-projective Dependency Parsing

Non-projective Dependency Parsing

Dependencies are allowed to cross

Example

Methods

- Nivre et al. (2004)
- Yamada and Matsumoto (2003)
- McDonald et al. (2005)

McDonald et al. (2005)

McDonald et al. (2005)

- State-of-the-art non-projective dependency parser.
- Based on finding the maximum spanning tree.
- Attachment decisions made independently.

McDonald et al. (2005)

McDonald et al. (2005)

- State-of-the-art non-projective dependency parser.
- Based on finding the maximum spanning tree.
- Attachment decisions made independently.

Example Mistake on Alpino Corpus

root	Ik	kom	om	twaalf	en	dan	krijg	je	wat	je	verdient
	I'll	come	at	twelve	and	then	get	you	what	you	deserve

McDonald et al. (2005)

McDonald et al. (2005)

- State-of-the-art non-projective dependency parser.
- Based on finding the maximum spanning tree.
- Attachment decisions made independently.

Example Mistake on Alpino Corpus

root	Ik	kom	om	twaalf	en	dan	krijg	je	wat	je	verdient
	I'll	come	at	twelve	and	then	get	you	what	you	deserve

McDonald and Pereira, 2006

- Second order scores
- *Approximate* search

More General

Chu-Liu-Edmonds, CYK, Viterbi

- More local models
- Optimality guaranteed
- Polynomial runtime guaranteed

More General

Chu-Liu-Edmonds, CYK, Viterbi

- More local models
- Optimality guaranteed
- Polynomial runtime guaranteed

Beam Search, Sampling

- More global models
- Optimality not guaranteed
- Polynomial runtime guaranteed

More General

Chu-Liu-Edmonds, CYK, Viterbi

- More local models
- Optimality guaranteed
- Polynomial runtime guaranteed

Beam Search, Sampling

- More global models
- Optimality not guaranteed
- Polynomial runtime guaranteed

Incremental ILP

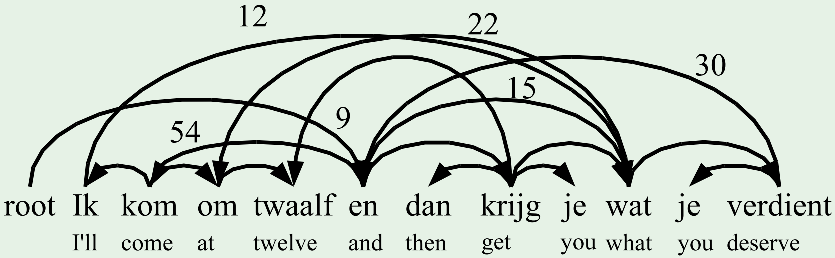
- More global models
- Optimality guaranteed
- Polynomial runtime not guaranteed

Overview

Outline

Maximum Spanning Tree Problem

Example Graph with Scores



MST Objective

Find the tree with the maximum sum of scores

More Formal

Score

$$s(\mathbf{x}, \mathbf{y}) = \sum_{(i,j,l)\in\mathbf{y}} s(i,j,l) = \sum_{(i,j,l)\in\mathbf{y}} \mathbf{w} \cdot \mathbf{f}(i,j,l)$$

for graph $\mathbf{y}$

More Formal

Score

$$s(\mathbf{x},\mathbf{y}) = \sum_{(i,j,l)\in\mathbf{y}} s(i,j,l) = \sum_{(i,j,l)\in\mathbf{y}} \mathbf{w}\cdot\mathbf{f}(i,j,l)$$

for graph $\mathbf{y}$

Constraint (Exactly One Head)

Exactly one head for each non-root token; no head for root

More Formal

Score

$$s(\mathbf{x},\mathbf{y}) = \sum_{(i,j,l)\in\mathbf{y}} s(i,j,l) = \sum_{(i,j,l)\in\mathbf{y}} \mathbf{w}\cdot\mathbf{f}(i,j,l)$$

for graph $\mathbf{y}$

Constraint (Exactly One Head)

Exactly one head for each non-root token; no head for root

Constraint (No Cycles)

The dependency graph contains no cycles

More Formal

Score

$$s(\mathbf{x},\mathbf{y}) = \sum_{(i,j,l)\in\mathbf{y}} s(i,j,l) = \sum_{(i,j,l)\in\mathbf{y}} \mathbf{w}\cdot\mathbf{f}(i,j,l)$$

for graph $\mathbf{y}$

Constraint (Exactly One Head)

Exactly one head for each non-root token; no head for root

Constraint (No Cycles)

The dependency graph contains no cycles

MST Objective

Maximise score under the two above constraints

Outline

Linguistic Constraints

Constraint

Coordination arguments must be compatible

Violated in

root	Ik	kom	om	twaalf	en	dan	krijg	je	wat	je	verdient
	I'll	come	at	twelve	and	then	get	you	what	you	deserve

Linguistic Constraints

Constraint

There must not be more than one subject for each verb

Violated in

subject subject

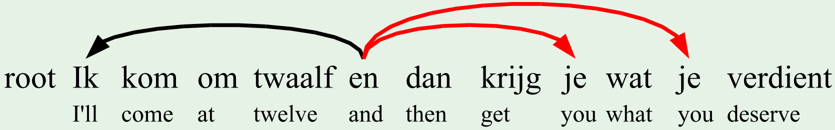
root	Ik	kom	om	twaalf	en	dan	krijg	je	wat	je	verdient
	I'll	come	at	twelve	and	then	get	you	what	you	deserve

Linguistic Constraints

Constraint

For each *and* coordination there is exactly one argument to the right and one more arguments to the left

Violated in



Outline

Outline

Decoding

Objective

Maximise:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{(i,j,l) \in \mathbf{y}} s(i, j, l)$$

given

- dependency parsing constraints
- linguistic constraints

Decoding

Objective

Maximise:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{(i,j,l) \in \mathbf{y}} s(i, j, l)$$

given

- dependency parsing constraints
- linguistic constraints

Methods

- Use the Chu-Liu-Edmonds algorithm (McDonald et al., 2005)
- Use some approximate search (McDonald and Pereira, 2006)
- Use Integer Linear Programming (Roth and Yih, 2005)

Decoding

Objective

Maximise:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{(i,j,l) \in \mathbf{y}} s(i, j, l)$$

given

- dependency parsing constraints
- linguistic constraints

Methods

- ~~Use the Chu-Liu-Edmonds algorithm (McDonald et al., 2005)~~
- Use some approximate search (McDonald and Pereira, 2006)
- Use Integer Linear Programming (Roth and Yih, 2005)

Decoding

Objective

Maximise:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{(i,j,l)\in\mathbf{y}} s(i,j,l)$$

given

- dependency parsing constraints
- linguistic constraints

Methods

- ~~Use the Chu-Liu-Edmonds algorithm (McDonald et al., 2005)~~
- ~~Use some approximate search (McDonald and Pereira, 2006)~~
- Use Integer Linear Programming (Roth and Yih, 2005)

Integer Linear Programming (ILP)

Decision Variables

$$x_1, x_2, x_3$$

Integer Linear Programming (ILP)

Decision Variables

$$x_1, x_2, x_3$$

Objective Function

$$1.5x_1 + 2x_2 - x_3$$

Integer Linear Programming (ILP)

Decision Variables

$$x_1, x_2, x_3$$

Objective Function

$$1.5x_1 + 2x_2 - x_3$$

Linear Constraints

$$\begin{aligned} x_1 + x_2 &< 2 \\ x_1 - x_3 &> 1 \end{aligned}$$

Integer Linear Programming (ILP)

Decision Variables

$$x_1, x_2, x_3$$

Objective Function

$$1.5x_1 + 2x_2 - x_3$$

Linear Constraints

$$\begin{aligned} x_1 + x_2 &< 2 \\ x_1 - x_3 &> 1 \end{aligned}$$

Integer Constraints

$$x_1 \in \{0, 1\}$$

Integer Linear Programming (ILP)

Decision Variables

$$x_1, x_2, x_3$$

Objective Function

$$1.5x_1 + 2x_2 - x_3$$

Linear Constraints

$$\begin{aligned} x_1 + x_2 &< 2 \\ x_1 - x_3 &> 1 \end{aligned}$$

Integer Constraints

$$x_1 \in \{0, 1\}$$

ILP Objective

Maximise objective function under constraints.

Integer Linear Programming (ILP)

Decision Variables

$$x_1, x_2, x_3$$

Objective Function

$$1.5x_1 + 2x_2 - x_3$$

Linear Constraints

$$\begin{aligned} x_1 + x_2 &< 2 \\ x_1 - x_3 &> 1 \end{aligned}$$

Integer Constraints

$$x_1 \in \{0, 1\}$$

ILP Objective

Maximise objective function under constraints.

Taskar 2004

Every Markov Network can be mapped to an polynomial-size ILP.

Dependency Parsing with ILP

Decision Variables

$$e_{i,j,l} = \begin{cases} 1 & \text{if there is a dependency from i to j with label l} \\ 0 & \text{otherwise} \end{cases}$$

for each token i, j and label l

Objective Function

$$\sum_{i,j,l} s(i,j,l) \cdot e_{i,j,l}$$

Dependency Parsing with ILP (2)

Auxiliary Variables

$$d_{i,j} = \begin{cases} 1 & \text{if there is a dependency from i to j} \\ 0 & \text{otherwise} \end{cases}$$

for each token i, j

Dependency Parsing with ILP (2)

Auxiliary Variables

$$d_{i,j} = \begin{cases} 1 & \text{if there is a dependency from i to j} \\ 0 & \text{otherwise} \end{cases}$$

for each token i, j

Only One Head

$$\sum_i d_{i,j} = 1$$

for all $j > 0$.

Dependency Parsing with ILP (2)

Auxiliary Variables

$$d_{i,j} = \begin{cases} 1 & \text{if there is a dependency from i to j} \\ 0 & \text{otherwise} \end{cases}$$

for each token i, j

Only One Head

$$\sum_i d_{i,j} = 1$$

for all $j > 0$.

No Cycles

$$\sum_{(i,j) \in G_s} d_{i,j} \leq |s|$$

for possible subsets all sets s of tokens

Dependency Parsing with ILP (2)

Auxiliary Variables

$$d_{i,j} = \begin{cases} 1 & \text{if there is a dependency from i to j} \\ 0 & \text{otherwise} \end{cases}$$

for each token i, j

Only One Head

$$\sum_i d_{i,j} = 1$$

for all $j > 0$.

No Cycles

$$\sum_{(i,j) \in G_s} d_{i,j} \leq |s|$$

for possible subsets all sets s of tokens

Dependency Parsing with ILP (2)

Auxiliary Variables

$$d_{i,j} = \begin{cases} 1 & \text{if there is a dependency from i to j} \\ 0 & \text{otherwise} \end{cases}$$

for each token i, j

Only One Head

$$\sum_i d_{i,j} = 1$$

for all $j > 0$.

No Cycles

$$\sum_{(i,j) \in G_s} d_{i,j} \leq |s|$$

for possible subsets all sets s of tokens

Germann et al. (2001)

Same cycle problem in ILP formulation for MT

Outline

Incremental Integer Linear Programming

Setup

- *base* (e.g. exactly one head) constraints
- *incremental* (e.g. no cycles) constraints

Incremental Integer Linear Programming

Setup

- *base* (e.g. exactly one head) constraints
- *incremental* (e.g. no cycles) constraints

Algorithm (see Warme (2002))

Set up ILP I with objective function and *base* constraints
repeat
  Solve I
  Find violated *incremental* constraints
  Add constraints to I
until No more constraints violated

Incremental Integer Linear Programming

Setup

- *base* (e.g. exactly one head) constraints
- *incremental* (e.g. no cycles) constraints

Algorithm (see Warme (2002))

Set up ILP I with objective function and *base* constraints
repeat
  Solve I
  Find violated *incremental* constraints
  Add constraints to I
until No more constraints violated

Tromble and Eisner (2006)

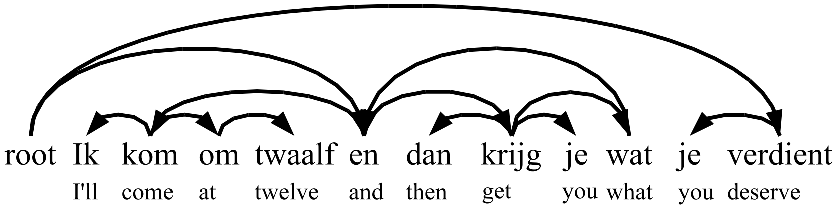
Replace ILP with finite-state automata

Outline

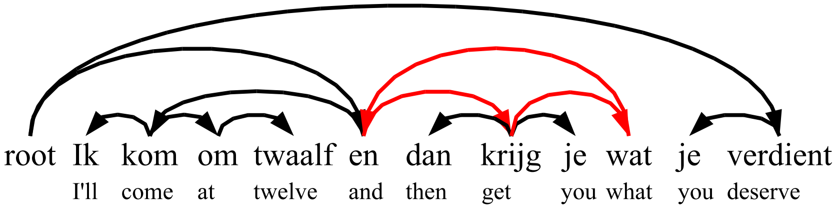
Example Sentence

root	Ik	kom	om	twaalf	en	dan	krijg	je	wat	je	verdient
	I'll	come	at	twelve	and	then	get	you	what	you	deserve

First Solution



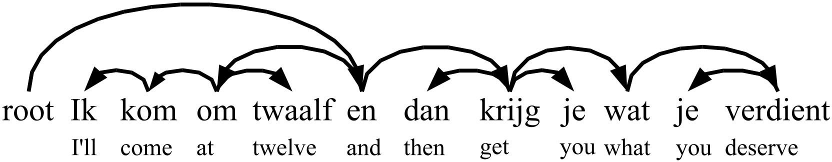
Add Violated Constraints



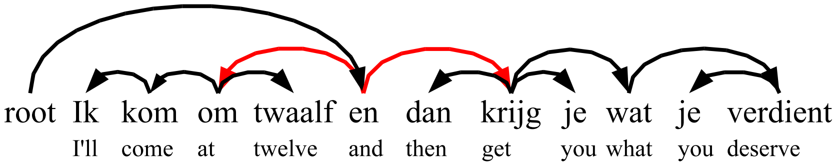
Add Constraint

$$d_{what,and} + d_{and,get} + d_{get,what} < 3$$

Next Solution



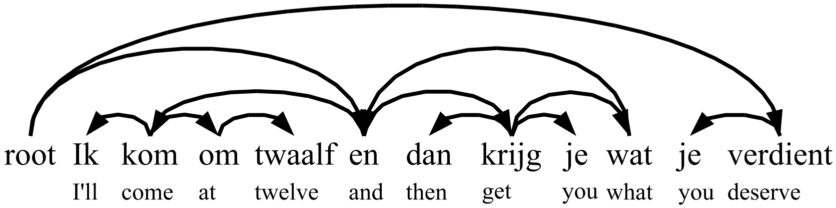
Add Violated Constraints



Add Constraint

$$d_{and,at} + d_{and,get} < 2$$

Done


root Ik kom om twaalf en dan krijg je wat je verdient
I'll come at twelve and then get you what you deserve

Outline

Training

Online Learning

1. single-best MIRA
2. Chu-Liu-Edmonds for parsing (McDonald et al. 2005)
3. No constraints

Training

Online Learning

1. single-best MIRA
2. Chu-Liu-Edmonds for parsing (McDonald et al. 2005)
3. No constraints

Roth and Yih (2005)

Training without constraints can actually help

Outline

Experiments

Questions

- How accurate in comparison to McDonald et. al (2005)?
- How fast/slow?

Experiments

Questions

- How accurate in comparison to McDonald et. al (2005)?
- How fast/slow?

Data

- Dutch alpino corpus from the CoNLL shared task 2006
- about 13000 Sentences, non-projective, 5% of edges crossing
- Split into development set and crossvalidation set
- Tuned feature and constraint set on dev set

Accuracy

In Comparison with McDonald et. al (2005)

Crossvalidation	Labelled	Unlabelled	Complete	Complete(U)
McDonald 2005	84.6%	88.9%	27.7%	42.2%
Incremental ILP	85.1%	89.4%	29.7%	43.8%

Statistical significant ($p < 0.001$ for Sign test and Dan Bikel's parse eval script)

Accuracy

In Comparison with McDonald et. al (2005)

Crossvalidation	Labelled	Unlabelled	Complete	Complete(U)
McDonald 2005	84.6%	88.9%	27.7%	42.2%
Incremental ILP	85.1%	89.4%	29.7%	43.8%

Statistical significant ($p < 0.001$ for Sign test and Dan Bikel's parse eval script)

With Others

- Wins on CoNLL test set but not significantly better than McDonald et al. (2006)
- Similar to performance of Malouf and van Noord (2004) (84.4% , smaller training set, evaluates control relations)

Runtime Evaluation

Exact Inference

- reasonable fast (0.5s for sentences with length between 20 - 30 tokens)
- significantly slower than McDonald et al. (2005) (3ms!)
- 150 times slower when parsing the full corpus (50min vs 20s) without feature extraction
- 6 times slower with feature extraction (add 10 minutes)
- 2 times slower with nearly loss less approximation method (see paper)

Outline

Conclusion

Accuracy

- Significantly better than McDonald et. al (2005)
- headroom left - rule engineering

Conclusion

Accuracy

- Significantly better than McDonald et. al (2005)
- headroom left - rule engineering

Runtime

- Significantly slower than McDonald et. al (2005)
- Feature extraction dominates runtime
- almost loss-less approximation available

Conclusion

Accuracy

- Significantly better than McDonald et. al (2005)
- headroom left - rule engineering

Runtime

- Significantly slower than McDonald et. al (2005)
- Feature extraction dominates runtime
- almost loss-less approximation available

In General

- Allows global models
- Guarantees optimality
- No polynomial runtime guarantee
- Good scores - fast processing

Future Work

Parsing

- 2nd order features
- Evaluate on more languages
- Joint POS tagging and parsing
- Joint constituent and dependency parsing

General

- Other applications(Collective IE, MT?)
- Generalise to features/potentials (as opposed to constraints)
- Theoretical runtime estimates

Thank you