Lagrangian Based Approaches for Lexicalized Tree Adjoining Grammar Parsing

Caio Corro

Supervision: Adeline Nazarenko & Joseph Le Roux

9 march 2018

Syntax: description of structures in natural languages

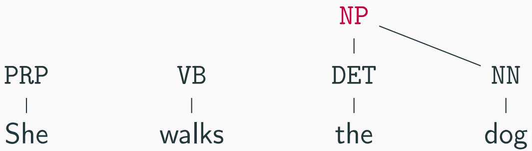
She walks the dog

PRP	VB	DET	NN
She	walks	the	dog

Syntactic analysis

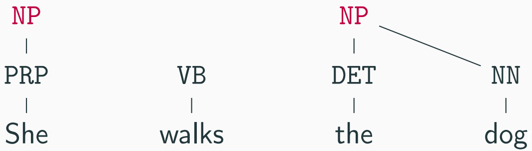
- **Part-of-speech tagging:** assign a category to each a lexical item

Syntax: description of structures in natural languages



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units

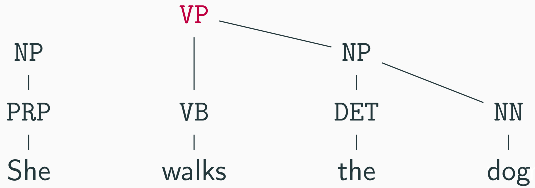


Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units

Syntactic analysis

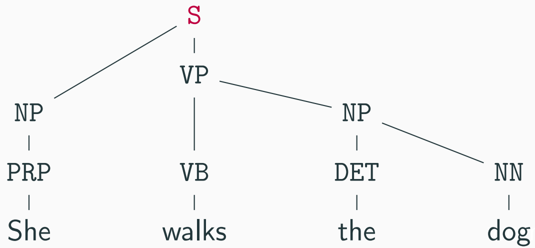
- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units

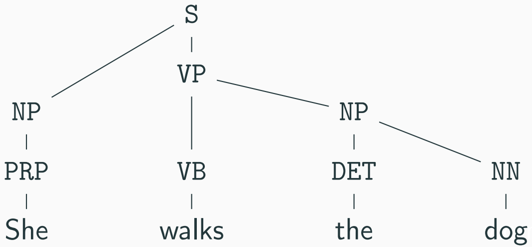
Syntax: description of structures in natural languages



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units

Syntax: description of structures in natural languages



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units

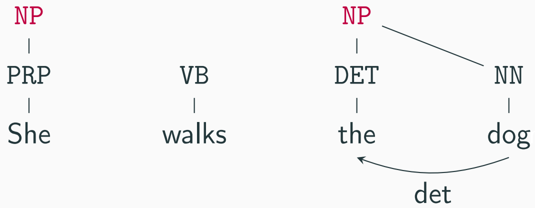
Syntax: description of structures in natural languages

PRP	VB	DET	NN
She	walks	the	dog

Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units
- **Dependency parsing:** define bi-lexical relations

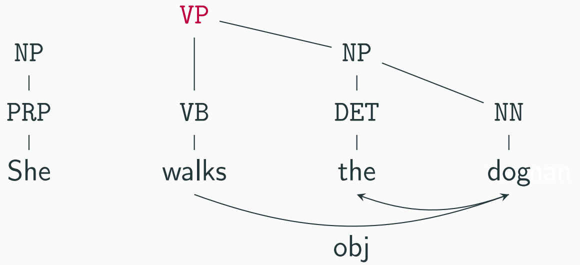
Syntax: description of structures in natural languages



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units
- **Dependency parsing:** define bi-lexical relations

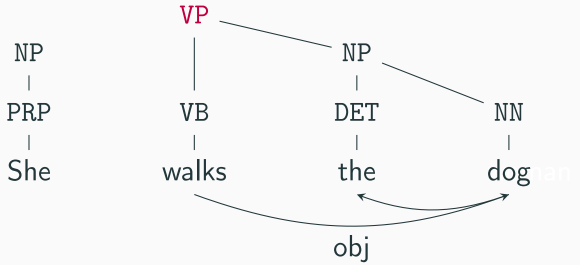
Syntax: description of structures in natural languages



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units
- **Dependency parsing:** define bi-lexical relations

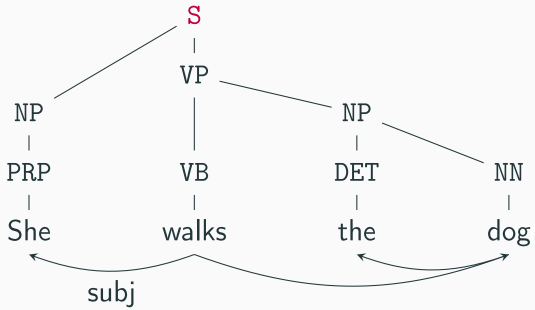
Syntax: description of structures in natural languages



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units
- **Dependency parsing:** define bi-lexical relations

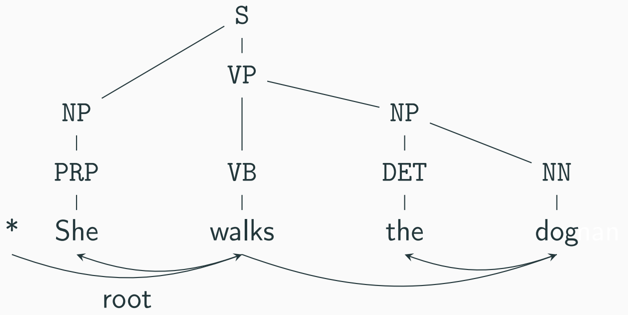
Syntax: description of structures in natural languages



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units
- **Dependency parsing:** define bi-lexical relations

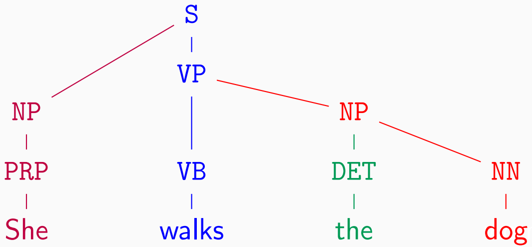
Syntax: description of structures in natural languages



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units
- **Dependency parsing:** define bi-lexical relations

Syntax: description of structures in natural languages



Syntactic analysis

- **Part-of-speech tagging:** assign a category to each a lexical item
- **Constituency parsing:** define a hierarchy of syntactic units
- **Dependency parsing:** define bi-lexical relations

Syntactic parsing

Parsing problem

Compute the best syntactic analysis for a given sentence

- **Input:** sentence
- **Output:** constituency/dependency structure

Usual algorithmic trade-off

- Exhaustive search with optimality certificate (dynamic program, ...)
- Heuristic without quality certificate (greedy/beam search, ...)

Syntactic parsing

Parsing problem

Compute the best syntactic analysis for a given sentence

- **Input:** sentence
- **Output:** constituency/dependency structure

Usual algorithmic trade-off

- Exhaustive search with optimality certificate (dynamic program, ...)
- Heuristic without quality certificate (greedy/beam search, ...)

Lagrangian relaxation

- Heuristic with quality/optimality certificate
- Guided exhaustive search

Scientific context

Syntactic analysis

Lexicalized Tree adjoining Grammar

- (Rich) tags
- Constituency structure
- Bi-lexical relations

Syntactic analysis

Lexicalized Tree adjoining Grammar

- (Rich) tags
- Constituency structure
- Bi-lexical relations

Weighted grammar

- Disambiguation

Scientific context

Syntactic analysis

Lexicalized Tree adjoining Grammar

- (Rich) tags
- Constituency structure
- Bi-lexical relations

Weighted grammar

- Disambiguation
- Robustness

Scientific context

Syntactic analysis

Lexicalized Tree adjoining Grammar

- (Rich) tags
- Constituency structure
- Bi-lexical relations

Weighted grammar

- Disambiguation
- Robustness

Parsing complexity

- $\mathcal{O}(n^7)$ with n the sentence length

Scientific context

Syntactic analysis

$\downarrow$

Graph theory

Benefits of reduction

- Alternative approach to problems
- Bottleneck characterization
- Substantial literature

Examples

- Dependency parsing
  $\Leftrightarrow$ Spanning Arborescence
  [McDonald et al. 2005]
- Translation
  $\Leftrightarrow$ Travelling Salesman Problem
  [Zaslavskiy et al. 2009]

Scientific context

Syntactic analysis

↓

Graph theory

↓

Integer Linear Programming

Declarative formulation

- y: syntactic structure
- $f(y)$: likelihood of the structure
- $g_i(y) \leq 0$: constraints on the structure

Scientific context

Syntactic analysis

$\downarrow$

Graph theory

$\downarrow$

Integer Linear Programming

Declarative formulation

- y: syntactic structure
- $f(y)$: likelihood of the structure
- $g_i(y) \leq 0$: constraints on the structure

Integer Linear Program

$$\begin{aligned} \max_y \quad & f(y) \\ \text{s.t.} \quad & g_i(y) \leq 0 \qquad \forall 1 \leq i \leq k \end{aligned}$$

Syntactic analysis

$\downarrow$

Graph theory

$\downarrow$

Integer Linear Programming

Declarative formulation

- y: syntactic structure
- $f(y)$: likelihood of the structure
- $g_i(y) \leq 0$: constraints on the structure

Integer Linear Program

$$\begin{aligned} \max_{y} \quad & f(y) \\ \text{s.t.} \quad & g_i(y) \leq 0 \qquad \forall 1 \leq i \leq k \end{aligned}$$

NLP Examples

Dependency parsing, model combination, semantic parsing . . .

[Rush et al. 2010; Koo et al. 2010; Le Roux et al. 2013; Das et al. 2012]

Scientific context

Syntactic analysis

$\downarrow$

Graph theory

$\downarrow$

Integer Linear Programming

$\downarrow$

Lagrangian relaxation

Difficult problem

$$\begin{aligned} \max_y \quad & f(y) \\ \text{s.t.} \quad & g_i(y) \leq 0 \qquad \forall 1 \leq i \leq k \\ & h_i(y) \leq 0 \qquad \forall 1 \leq i \leq l \end{aligned}$$

Intuition

Difficult problem because of $h_i(y)$
$\Rightarrow$ use soft penalties in the objective instead

Scientific context

Syntactic analysis

↓

Graph theory

↓

Integer Linear Programming

↓

Lagrangian relaxation

Difficult problem

$$\begin{aligned} \max_{y} \quad & f(y) \\ \text{s.t.} \quad & g_i(y) \leq 0 \qquad \forall 1 \leq i \leq k \\ & h_i(y) \leq 0 \qquad \forall 1 \leq i \leq l \end{aligned}$$

Intuition

Difficult problem because of $h_i(y)$
$\Rightarrow$ use soft penalties in the objective instead

What we get

- Bounds on the original problem
- Possibly an optimality certificate

Scientific context

Syntactic analysis

LTAG derivation tree parsing

Joint tagging and parsing

Scientific context

Syntactic analysis	LTAG derivation tree parsing	Joint tagging and parsing
↓	↓	↓
Graph theory	**YRMSA**	**GMSA**

Scientific context

Syntactic analysis

↓

Graph theory

↓

Integer Linear Programming

LTAG derivation tree parsing

↓

YRMSA

↓

Non-compact program

Joint tagging and parsing

↓

GMSA

↓

Compact program

Scientific context

Syntactic analysis → Graph theory → Integer Linear Programming → **Lagrangian relaxation**

LTAG derivation tree parsing → YRMSA → Non-compact program → **Non delayed relax-and-cut**

Joint tagging and parsing → GMSA → Compact program → **Dual decomposition**

Outline

1. Lexicalized Tree Adjoining Grammar Parsing
2. Efficient parsing with Lagrangian relaxation
3. A dependency-like LTAG parser
4. Joint Tagging and Dependency Parsing
5. Conclusion

1. Lexicalized Tree Adjoining Grammar Parsing

Lexicalized Tree Adjoining Grammar (LTAG)

Motivations

- Mildly context-sensitive formalism
- Linguistically plausible
- Semantics

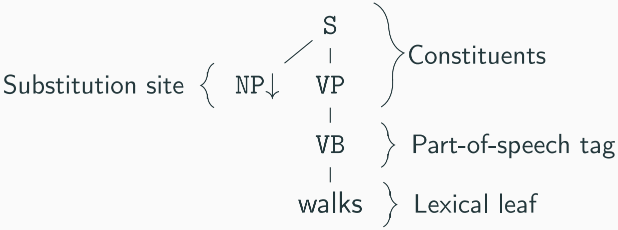
Lexicalized Tree Adjoining Grammar (LTAG)

Motivations

- Mildly context-sensitive formalism
- Linguistically plausible
- Semantics

Elementary tree

Extended part-of-speech tags with structural constraints
e.g. *A verb with a subject on its left-side*



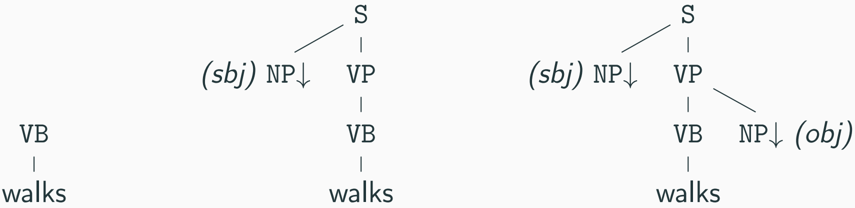
Example

VB

walks

(verb)

Example

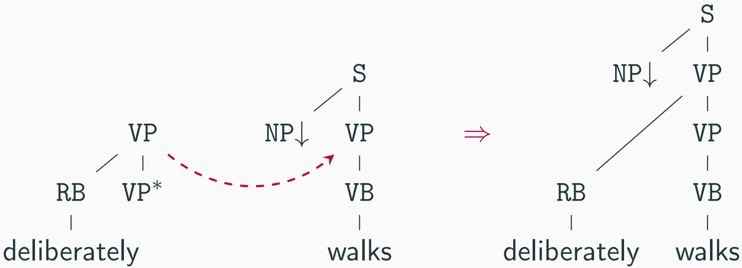


(verb) (intransitive verb) (transitive verb)

Substitution

Elementary tree combination

Substitution

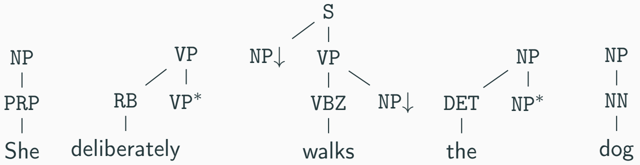


Adjunction

LTAG derivation tree

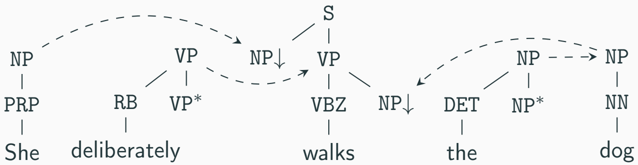
She deliberately walks the dog

Bottom-up construction of the syntactic phrase structure



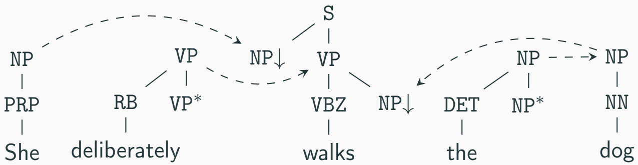
Bottom-up construction of the syntactic phrase structure

LTAG derivation tree

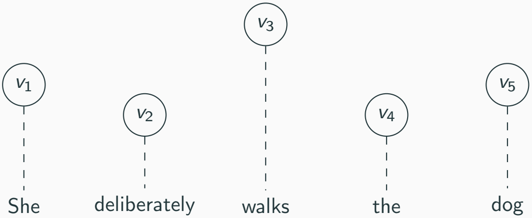


Bottom-up construction of the syntactic phrase structure

LTAG derivation tree

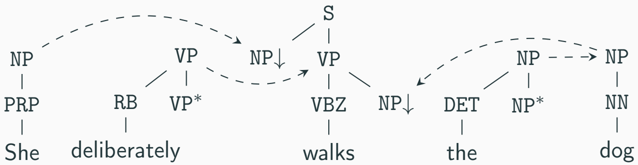


Bottom-up construction of the syntactic phrase structure

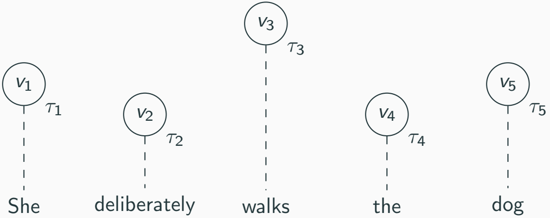


Representation alternative as a derivation tree [Rambow et al. 1997]

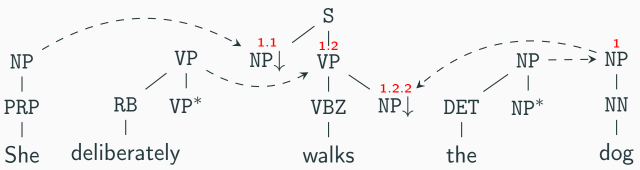
LTAG derivation tree



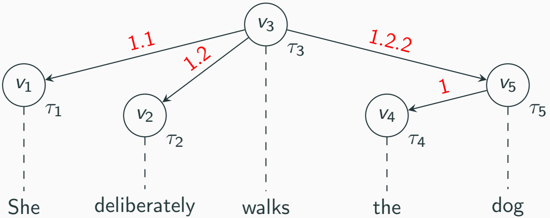
Bottom-up construction of the syntactic phrase structure



Representation alternative as a derivation tree [Rambow et al. 1997]



Bottom-up construction of the syntactic phrase structure



Representation alternative as a derivation tree [Rambow et al. 1997]

Weighted LTAG parsing

Weights

- Tag weights (elementary tree assignation)
- Dependency weights (combination operations)

Parsing goal

- Compute the syntactic structure of maximum weight

Weighted LTAG parsing

Weights

- Tag weights (elementary tree assignation)
- Dependency weights (combination operations)

Parsing goal

- Compute the syntactic structure of maximum weight

Complexity [Eisner et al. 2000]

$\mathcal{O}(n^6 \max(n, g) gt)$:

n: sentence length

t: maximum tree size

g: maximum ambiguity

$\Rightarrow \mathcal{O}(n^7)$ asymptotically w.r.t. the sentence length

2. Efficient parsing with Lagrangian relaxation

Integer Linear Programming

Integer Linear Program (ILP)

$$\max_{y} \quad y^\top w \qquad \text{(maximize the weight of the structure } y\text{)}$$

$$\text{s.t.} \quad Ay - b \leq 0 \qquad \text{(constraints on the structure)}$$

Integer Linear Programming

Integer Linear Program (ILP)

$$\begin{aligned} \max_y \quad & y^\top w && \text{(maximize the weight of the structure } y\text{)} \\ \text{s.t.} \quad & Ay - b \leq 0 && \text{(easy constraints)} \\ & By - c \leq 0 && \text{(difficult constraints)} \end{aligned}$$

Integer Linear Programming

Integer Linear Program (ILP)

$$\begin{aligned}
\max_{y} \quad & y^\top w && \text{(maximize the weight of the structure } y\text{)} \\
\text{s.t.} \quad & Ay - b \leq 0 && \text{(easy constraints)} \\
& By - c \leq 0 && \text{(difficult constraints)}
\end{aligned}$$

Intuition

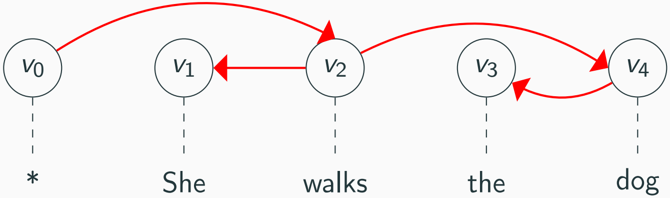
- Remove difficult constraints
- Introduce them as penalties in the objective
- Solve the new reparametrized problem iteratively

Example: dependency parsing

She walks the dog

Example: dependency parsing

Example: dependency parsing



Reduction

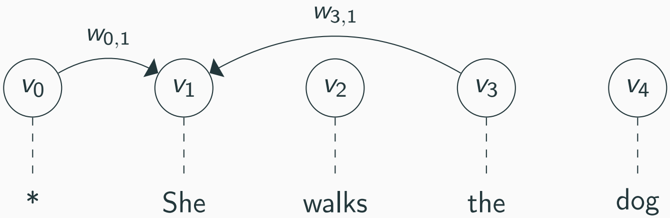
Dependency tree $\Leftrightarrow$ v_0-rooted spanning arborescence

i.e. a connected graph such that:

- v_0: no incoming arc
- $v_1 \ldots v_4$: exactly one incoming arc
- Acyclic

Graph construction

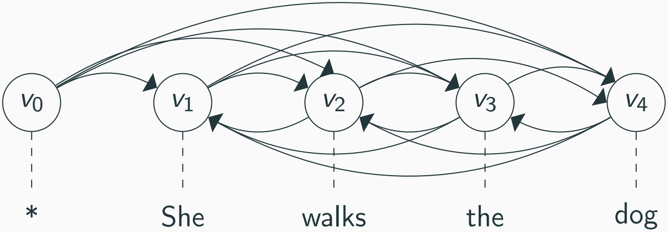
1. Add arc candidates
2. Add arc weights



Graph construction

1. Add arc candidates
2. Add arc weights

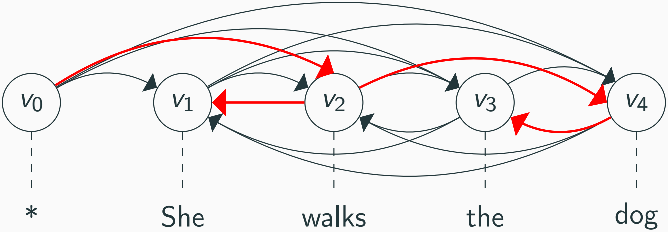
Example: dependency parsing



Graph construction

1. Add arc candidates
2. Add arc weights

Example: dependency parsing



Graph construction

1. Add arc candidates
2. Add arc weights
3. Compute the spanning arborescence of maximum weight

Example: dependency parsing

ILP formulation

- y: arc incidence vector ($y_a = 1$ iff arc a is selected)
- w: arc weight vector

$$\max_y \quad y^\top w \qquad \text{(arc-factored model)}$$

Example: dependency parsing

ILP formulation

- y: arc incidence vector ($y_a = 1$ iff arc a is selected)
- w: arc weight vector

$$\max_{y} \quad y^\top w \qquad \text{(arc-factored model)}$$

$$\text{s.t.} \quad \sum_{a \in \delta^{\text{in}}(v_0)} y_a = 0 \qquad \text{(root)}$$

Example: dependency parsing

ILP formulation

- y: arc incidence vector ($y_a = 1$ iff arc a is selected)
- w: arc weight vector

$$
\begin{aligned}
\max_{y} \quad & y^\top w && && \text{(arc-factored model)} \\
\text{s.t.} \quad & \sum_{a \in \delta^{\mathrm{in}}(v_0)} y_a = 0 && && \text{(root)} \\
& \sum_{a \in \delta^{\mathrm{in}}(v)} y_a = 1 && \forall v \in V^+ && \text{(one head/word)}
\end{aligned}
$$

Example: dependency parsing

ILP formulation

- y: arc incidence vector ($y_a = 1$ iff arc a is selected)
- w: arc weight vector

$$
\begin{aligned}
\max_{y} \quad & y^\top w && && \text{(arc-factored model)} \\
\text{s.t.} \quad & \sum_{a \in \delta^{\mathrm{in}}(v_0)} y_a = 0 && && \text{(root)} \\
& \sum_{a \in \delta^{\mathrm{in}}(v)} y_a = 1 && \forall v \in V^+ && \text{(one head/word)} \\
& \sum_{a \in \delta^{\mathrm{in}}(W)} y_a \geq 1 && \forall W \subseteq V^+ && \text{(connectedness)}
\end{aligned}
$$

Example: dependency parsing

ILP formulation

- y: arc incidence vector ($y_a = 1$ iff arc a is selected)
- w: arc weight vector

$$
\begin{aligned}
\max_{y} \quad & y^\top w & & \text{(arc-factored model)} \\
\text{s.t.} \quad & \sum_{a \in \delta^{\text{in}}(v_0)} y_a = 0 & & \text{(root)} \\
& \sum_{a \in \delta^{\text{in}}(v)} y_a = 1 & \forall v \in V^+ \quad & \text{(one head/word)} \\
& \sum_{a \in \delta^{\text{in}}(W)} y_a \geq 1 & \forall W \subseteq V^+ \quad & \text{(connectedness)} \\
& y \in \{0, 1\}^A & & \text{(integrality)}
\end{aligned}
$$

Example: dependency parsing

ILP formulation

- y: arc incidence vector ($y_a = 1$ iff arc a is selected)
- w: arc weight vector

$$\max_{y} \quad y^\top w \qquad \text{s.t.} \quad y \in \mathcal{Y}$$

Efficient decoding

- Generic solver: simplex, interior point method, ...
- Specialized algorithm: Maximum Spanning Arborescence $\mathcal{O}(n^2)$ [Edmonds 1967; Schrijver 2003; McDonald et al. 2005]

Lagrangian relaxation

Difficult constraints

Force each vertex to have at most k outgoing arc

Difficult constraints

Force each vertex to have at most k outgoing arc

ILP formulation

$$\begin{aligned} \max_{y} \quad & y^\top w \\ \text{s.t.} \quad & y \in \mathcal{Y} && \text{(easy constraints)} \end{aligned}$$

Lagrangian relaxation

Difficult constraints

Force each vertex to have at most k outgoing arc

ILP formulation

$$
\begin{aligned}
\max_{y} \quad & y^\top w \\
\text{s.t.} \quad & y \in \mathcal{Y} && \text{(easy constraints)} \\
& \sum_{a \in \delta^+(v)} y_a \leq k \qquad \forall v \in V && \text{(hard constraints)}
\end{aligned}
$$

Lagrangian relaxation

Difficult constraints

Force each vertex to have at most k outgoing arc

ILP formulation

$$\begin{aligned}
\max_y \quad & y^\top w \\
\text{s.t.} \quad & y \in \mathcal{Y} & & \text{(easy constraints)} \\
& \sum_{a \in \delta^+(v)} y_a \leq k \qquad \forall v \in V & & \text{(hard constraints)}
\end{aligned}$$

Lagrangian relaxation

1. Relax difficult constraints as penalties in the objective
   $\lambda \geq 0$: vector of Lagrangian multipliers

Lagrangian relaxation

Difficult constraints

Force each vertex to have at most k outgoing arc

Lagrangian dual

$$\begin{aligned} \max_{y} \quad & y^\top w - \sum_{v \in V} \lambda_v \Big(\sum_{a \in \delta^+ v} y_a - k \Big) \\ \text{s.t.} \quad & y \in \mathcal{Y} \qquad\qquad \text{(easy constraints)} \end{aligned}$$

Lagrangian relaxation

1. Relax difficult constraints as penalties in the objective
   $\lambda \geq 0$: vector of Lagrangian multipliers
   $\Rightarrow$ **Upper bound on the original problem**

Lagrangian relaxation

Difficult constraints

Force each vertex to have at most k outgoing arc

Lagrangian dual

$$\max_y \quad y^\top w' \quad (+\text{ constant term w.r.t. } y)$$
$$\text{s.t.} \quad y \in \mathcal{Y} \qquad\qquad \text{(easy constraints)}$$

Lagrangian relaxation

1. Relax difficult constraints as penalties in the objective
   $\lambda \geq 0$: vector of Lagrangian multipliers
2. Rewrite the objective

Lagrangian relaxation

Difficult constraints

Force each vertex to have at most k outgoing arc

Lagrangian dual

$$\min_{\lambda} \quad \max_{y} \quad y^\top w' \quad (+\text{ constant term w.r.t. } y)$$
$$\text{s.t.} \quad y \in \mathcal{Y} \qquad \text{(easy constraints)}$$

Lagrangian relaxation

1. Relax difficult constraints as penalties in the objective
   $\lambda \geq 0$: vector of Lagrangian multipliers
2. Rewrite the objective
3. Minimize over λ

Lagrangian optimization

Lagrangian dual

$$\min_{\lambda} \quad \max_{y} \quad y^\top w' \quad (+ \text{ constant term w.r.t. } y)$$
$$\text{s.t.} \quad y \in \mathcal{Y}$$

Optimization

- $\max_y$: easy (assumption) $\Rightarrow$ MSA
- $\min_\lambda$: subgradient descent $\Rightarrow$ loop over the maximization

Lagrangian optimization

Lagrangian dual

$$\min_{\lambda} \quad \max_{y} \quad y^\top w' \quad (+ \text{ constant term w.r.t. } y)$$
$$\text{s.t.} \quad y \in \mathcal{Y}$$

Optimization

- $\max_y$: easy (assumption) $\Rightarrow$ MSA
- $\min_\lambda$: subgradient descent $\Rightarrow$ loop over the maximization

Heuristic

- Quality certificate
- Possible optimality certificate

Exhaustive search

- Branch-and-bound
- Exact pruning

Methodology

1. Graph characterization of LTAG-derivations
2. ILP formulation of the problem
3. Lagrangian based decoder

Methodology

Methodology

1. Graph characterization of LTAG-derivations
2. ILP formulation of the problem
3. Lagrangian based decoder

Requirements for the ILP formulation

- Formulation as linear inequalities

Methodology

1. Graph characterization of LTAG-derivations
2. ILP formulation of the problem
3. Lagrangian based decoder

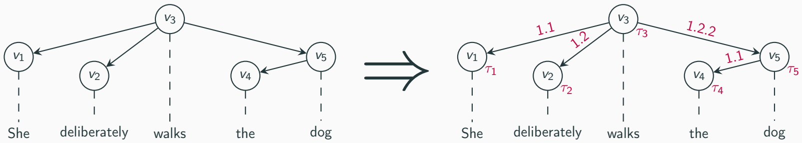
Requirements for the ILP formulation

- Formulation as linear inequalities

Requirements for the Lagrangian based decoder

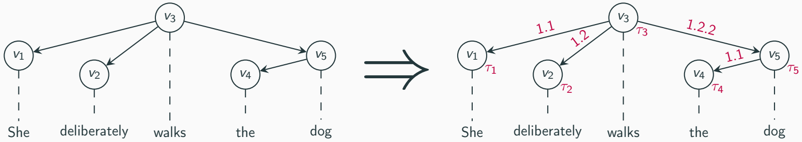
- Relaxation with "nice" objective function
- Efficient algorithm that solve the relaxed problem

3. A dependency-like LTAG parser



A dependency-like LTAG parser

1. LTAG compatible dependency parsing [Corro et al. 2016]
2. LTAG derivation tree labeler [Corro et al. 2017b]



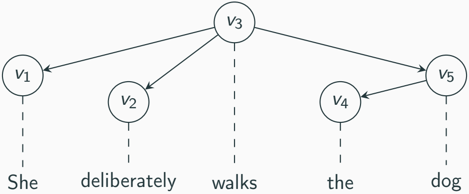
A dependency-like LTAG parser

1. LTAG compatible dependency parsing [Corro et al. 2016]
2. LTAG derivation tree labeler [Corro et al. 2017b]

Methodology

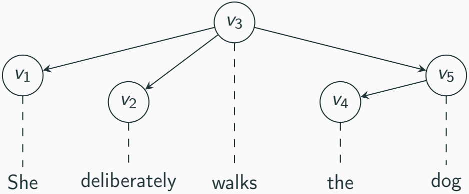
1. Graph characterization of LTAG-derivations
2. ILP formulation of the problem
3. Lagrangian based decoder

Dependency trees



Structural properties of dependency structures

Non-projective $\longleftrightarrow$ **Projective**



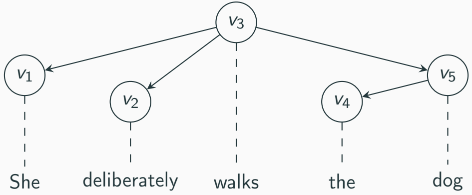
Structural properties of dependency structures

k-Bounded Block Degree

Non-projective Projective

Well-nestedness

Dependency trees



Structural properties of dependency structures

Non-projective ↔ **k-Bounded Block Degree** ↔ Projective

Non-projective ↔ **Well-nestedness** ↔ Projective

LTAG derivation tree [Bodirsky et al. 2009; Kuhlmann 2010]

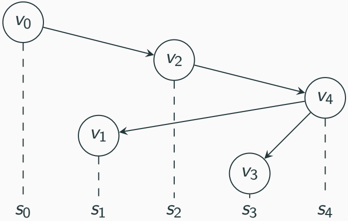
- 2-Bounded Block Degree (2-BBD)
- Well-nested (WN)

Yield

Yield of a vertex v

Set of all vertices reachable from v

$\Rightarrow$ Required in order to defined structural properties

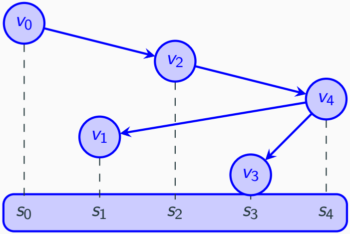


Yield

Yield of a vertex v

Set of all vertices reachable from v

$\Rightarrow$ Required in order to defined structural properties

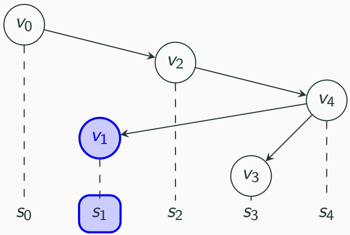


$Yield(v_0) = \{v_0, v_1, v_2, v_3, v_4\}$

Yield of a vertex v

Set of all vertices reachable from v

$\Rightarrow$ Required in order to defined structural properties



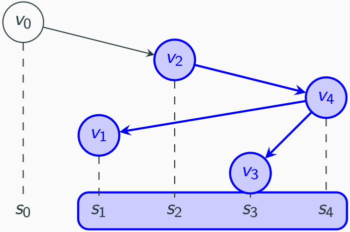
$Yield(v_0) = \{v_0, v_1, v_2, v_3, v_4\}$

$Yield(v_1) = \{v_1\}$

Yield of a vertex v

Set of all vertices reachable from v

$\Rightarrow$ Required in order to defined structural properties



$Yield(v_0) = \{v_0, v_1, v_2, v_3, v_4\}$

$Yield(v_1) = \{v_1\}$

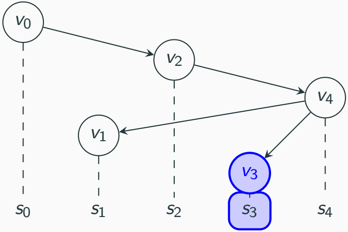
$Yield(v_2) = \{v_1, v_2, v_3, v_4\}$

Yield

Yield of a vertex v

Set of all vertices reachable from v

$\Rightarrow$ Required in order to defined structural properties



$Yield(v_0) = \{v_0, v_1, v_2, v_3, v_4\}$

$Yield(v_1) = \{v_1\}$

$Yield(v_2) = \{v_1, v_2, v_3, v_4\}$

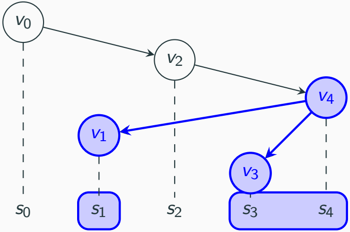
$Yield(v_3) = \{v_3\}$

Yield

Yield of a vertex v

Set of all vertices reachable from v

$\Rightarrow$ Required in order to defined structural properties



$Yield(v_0) = \{v_0, v_1, v_2, v_3, v_4\}$

$Yield(v_1) = \{v_1\}$

$Yield(v_2) = \{v_1, v_2, v_3, v_4\}$

$Yield(v_3) = \{v_3\}$

$Yield(v_4) = \{v_3, v_4\}$

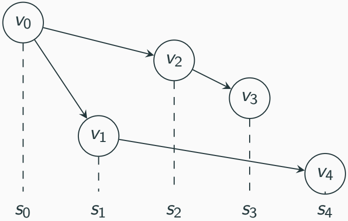
Contiguous yield

Block degree of a vertex

Minimum number of intervals needed to describe its yield

Contiguous yield

Yield which can be defined with a single interval

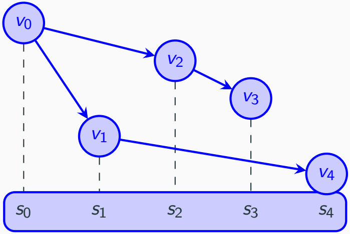


Block degree of a vertex

Minimum number of intervals needed to describe its yield

Contiguous yield

Yield which can be defined with a single interval



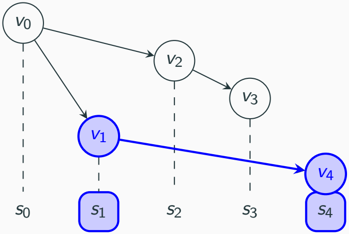
$Yield(v_0) = [v_0 \dots v_4] \qquad BD(v_0) = 1$

Block degree of a vertex

Minimum number of intervals needed to describe its yield

Contiguous yield

Yield which can be defined with a single interval



$Yield(v_0) = [v_0 \ldots v_4] \qquad BD(v_0) = 1$

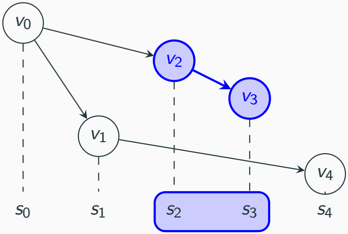
$Yield(v_1) = [v_1] \cup [v_4] \qquad BD(v_1) = 2$

Block degree of a vertex

Minimum number of intervals needed to describe its yield

Contiguous yield

Yield which can be defined with a single interval



$$Yield(v_0) = [v_0 \dots v_4] \qquad BD(v_0) = 1$$

$$Yield(v_1) = [v_1] \cup [v_4] \qquad BD(v_1) = 2$$

$$Yield(v_2) = [v_2 \dots v_3] \qquad BD(v_2) = 1$$

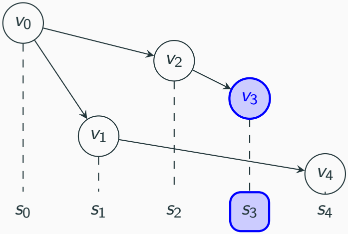
Contiguous yield

Block degree of a vertex

Minimum number of intervals needed to describe its yield

Contiguous yield

Yield which can be defined with a single interval



$$Yield(v_0) = [v_0 \dots v_4] \qquad BD(v_0) = 1$$

$$Yield(v_1) = [v_1] \cup [v_4] \qquad BD(v_1) = 2$$

$$Yield(v_2) = [v_2 \dots v_3] \qquad BD(v_2) = 1$$

$$Yield(v_3) = [v_3] \qquad BD(v_3) = 1$$

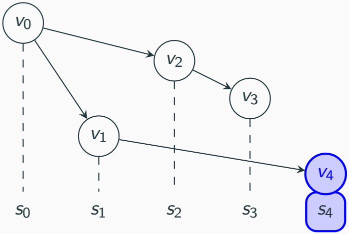
Contiguous yield

Block degree of a vertex

Minimum number of intervals needed to describe its yield

Contiguous yield

Yield which can be defined with a single interval



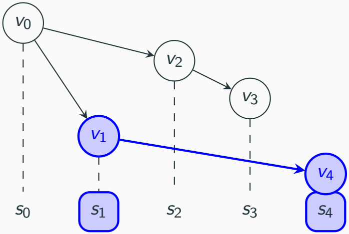
$$
\begin{aligned}
&Yield(v_0) = [v_0 \dots v_4] && BD(v_0) = 1 \\
&Yield(v_1) = [v_1] \cup [v_4] && BD(v_1) = 2 \\
&Yield(v_2) = [v_2 \dots v_3] && BD(v_2) = 1 \\
&Yield(v_3) = [v_3] && BD(v_3) = 1 \\
&Yield(v_4) = [v_4] && BD(v_4) = 1
\end{aligned}
$$

Block degree of a vertex

Minimum number of intervals needed to describe its yield

Contiguous yield

Yield which can be defined with a single interval

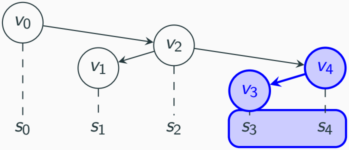


$Yield(v_0) = [v_0 \dots v_4]$	$BD(v_0) = 1$
$Yield(v_1) = [v_1] \cup [v_4]$	$BD(v_1) = 2$
$Yield(v_2) = [v_2 \dots v_3]$	$BD(v_2) = 1$
$Yield(v_3) = [v_3]$	$BD(v_3) = 1$
$Yield(v_4) = [v_4]$	$BD(v_4) = 1$

Structural properties of dependency trees

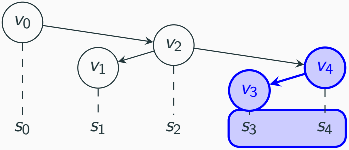
Projective dependency tree

Arborescence with contiguous yields only



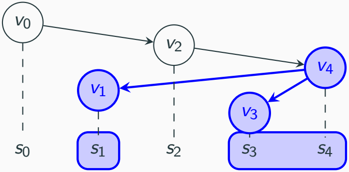
Projective dependency tree

Arborescence with contiguous yields only



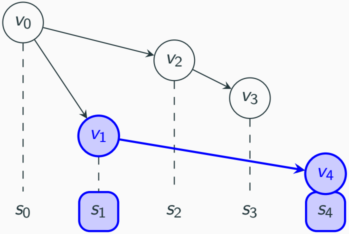
Non-projective dependency tree

Arborescence with at least one non-contiguous yield



k-Bounded Block Degree (k-BBD)

- BD of a tree: the maximal block degree of its vertices
- k-BBD tree: tree with a BD less or equal to k



$Yield(v_0) = [v_0 \dots v_4]$ $\quad BD(v_0) = 1$

$Yield(v_1) = [v_1] \cup [v_4]$ $\quad BD(v_1) = 2$

$Yield(v_2) = [v_2 \dots v_3]$ $\quad BD(v_2) = 1$

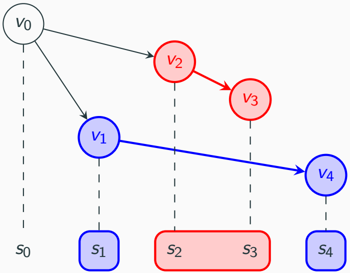
$Yield(v_3) = [v_3]$ $\quad BD(v_3) = 1$

$Yield(v_4) = [v_4]$ $\quad BD(v_4) = 1$

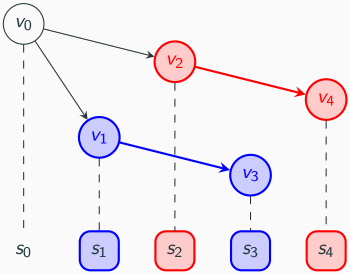
Tree of block degree 2

Well-nestedness (WN)

- Interleaving sets $I_1 \cap I_2 = \emptyset$:
  $\exists i, j \in I_1$ and $k, l \in I_2$ such that $i < k < j < l$
- Well-nested tree: does not contain two vertices whose yields interleave
  $\Rightarrow$ e.g. a yield cannot be inside and outside a gap



Well-nested tree



Not well-nested tree

Complexity

Non-projective	$\mathcal{O}(n^2)$	[McDonald et al. 2005]
Projective	$\mathcal{O}(n^3)$	[Eisner 2000]
WN + 2-BBD	$\mathcal{O}(n^7)$	[Gómez-Rodríguez et al. 2009]
WN + k-BBD, $k \geq 2$	$\mathcal{O}(n^{5+2(k-1)})$	[Gómez-Rodríguez et al. 2009]
k-BBD, $k \geq 2$	NP-complete	[Satta 1992]

Remark

Same complexity as LTAG parsing :(

Contribution

- ILP formulation of the problem
- Solver based on Lagrangian relaxation

k-Bounded Block Degree Constraint

Definition

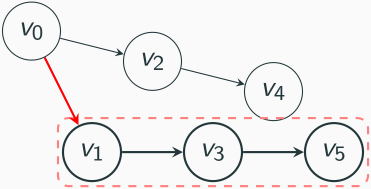
$\mathcal{W}^{\geq k+1}$: vertex subsets describing at least $k+1$ intervals

k-Bounded Block Degree Constraint

Definition

$\mathcal{W}^{\geq k+1}$: vertex subsets describing at least $k+1$ intervals

Example with $k = 2$ **and** $[v_1] \cup [v_3] \cup [v_5] \in \mathcal{W}^{\geq 3}$



Not 2-BBD

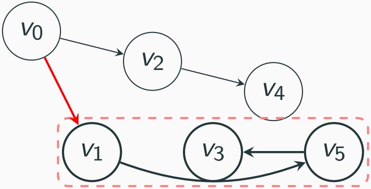
$\rightarrow$: incoming/outgoing arcs to the vertex subset $[v_1] \cup [v_3] \cup [v_5]$

k-Bounded Block Degree Constraint

Definition

$\mathcal{W}^{\geq k+1}$: vertex subsets describing at least $k+1$ intervals

Example with $k = 2$ **and** $[v_1] \cup [v_3] \cup [v_5] \in \mathcal{W}^{\geq 3}$



Not 2-BBD

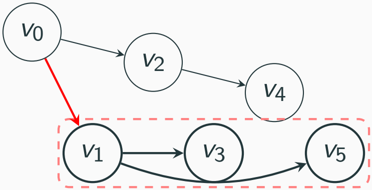
$\rightarrow$: incoming/outgoing arcs to the vertex subset $[v_1] \cup [v_3] \cup [v_5]$

k-Bounded Block Degree Constraint

Definition

$\mathcal{W}^{\geq k+1}$: vertex subsets describing at least $k+1$ intervals

Example with $k = 2$ and $[v_1] \cup [v_3] \cup [v_5] \in \mathcal{W}^{\geq 3}$



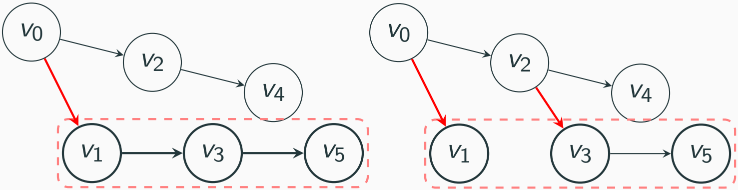
Not 2-BBD

$\rightarrow$: incoming/outgoing arcs to the vertex subset $[v_1] \cup [v_3] \cup [v_5]$

Definition

$\mathcal{W}^{\geq k+1}$: vertex subsets describing at least $k+1$ intervals

Example with $k = 2$ **and** $[v_1] \cup [v_3] \cup [v_5] \in \mathcal{W}^{\geq 3}$



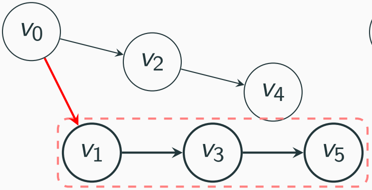
Not 2-BBD   2-BBD

$\rightarrow$: incoming/outgoing arcs to the vertex subset $[v_1] \cup [v_3] \cup [v_5]$

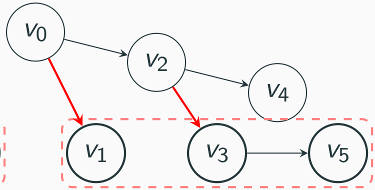
Definition

$\mathcal{W}^{\geq k+1}$: vertex subsets describing at least $k+1$ intervals

Example with $k = 2$ **and** $[v_1] \cup [v_3] \cup [v_5] \in \mathcal{W}^{\geq 3}$



Not 2-BBD



2-BBD

Constraint

$\forall W \in \mathcal{W}^{\geq k+1} \Rightarrow$ At least two incoming/outgoing arcs

Well-nestedness constraint

Notation

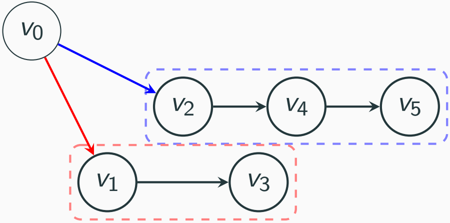
$\mathcal{I}$: family of pairs of disjoint interleaving vertex subsets

Well-nestedness constraint

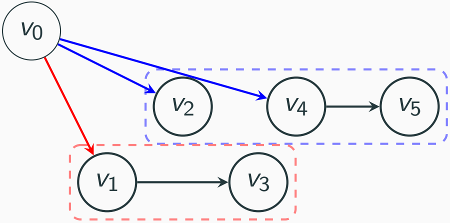
Notation

$\mathcal{I}$: family of pairs of disjoint interleaving vertex subsets

Example with $(\{1,3\},\{2,4,5\}) \in \mathcal{I}$



Not Well-nested



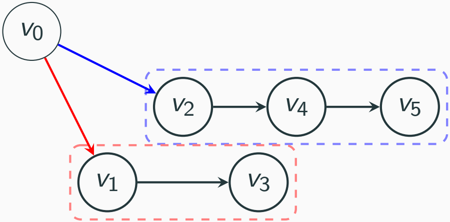
Well-nested

Well-nestedness constraint

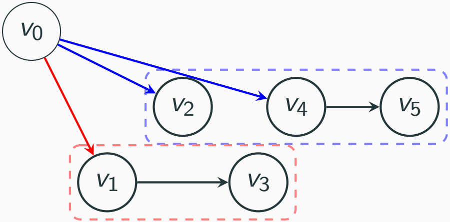
Notation

$\mathcal{I}$: family of pairs of disjoint interleaving vertex subsets

Example with $(\{1, 3\}, \{2, 4, 5\}) \in \mathcal{I}$



Not Well-nested



Well-nested

Constraint

For each couple $(I_1, I_2) \in \mathcal{I}$

$\Rightarrow$ At least two incoming/outgoing arcs for I_1 or I_2

Full ILP: parsing with k-BBD and WN constraints

$$
\begin{aligned}
\max_{y} \quad & y^\top w & & \text{(Arc-factored)} \\
\text{s.t.} \quad & y \in Y & & \text{(Arborescence)} \\
& \sum_{a \in \delta(W)} y_a \geq 2 & \forall\ W \in \mathcal{W}^{\geq k+1} \quad & \text{(k-BBD)} \\
& \sum_{a \in \delta(I_1)} y_a + \sum_{a \in \delta(I_2)} y_a \geq 3 & \forall (I_1, I_2) \in \mathcal{I} \quad & \text{(WN)}
\end{aligned}
$$

Full ILP: parsing with k-BBD and WN constraints

$$
\begin{aligned}
\max_{y} \quad & y^\top w && && \text{(Arc-factored)} \\
\text{s.t.} \quad & y \in Y && && \text{(Arborescence)} \\
& \sum_{a \in \delta(W)} y_a \geq 2 && \forall\ W \in \mathcal{W}^{\geq k+1} && \text{(k-BBD)} \\
& \sum_{a \in \delta(I_1)} y_a + \sum_{a \in \delta(I_2)} y_a \geq 3 && \forall (I_1, I_2) \in \mathcal{I} && \text{(WN)}
\end{aligned}
$$

Problem

- MSA: k-BBD and WN constraints can not be integrated
- Generic solver: exponential number of constraints
- No efficient algorithm [Gómez-Rodríguez et al. 2009]

Full ILP: parsing with k-BBD and WN constraints

$$
\begin{aligned}
\max_{y} \quad & y^\top w && && \text{(Arc-factored)}\\
\text{s.t.} \quad & y \in Y && && \text{(Arborescence)}\\
& \sum_{a \in \delta(W)} y_a \geq 2 && \forall\, W \in \mathcal{W}^{\geq k+1} && \text{(k-BBD)}\\
& \sum_{a \in \delta(I_1)} y_a + \sum_{a \in \delta(I_2)} y_a \geq 3 && \forall (I_1, I_2) \in \mathcal{I} && \text{(WN)}
\end{aligned}
$$

Problem

- MSA: k-BBD and WN constraints can not be integrated
- Generic solver: exponential number of constraints
- No efficient algorithm [Gómez-Rodríguez et al. 2009]

Solving the ILP
$\Rightarrow$ Lagrangian Relaxation applied on k-BBD/WN constraints

Lagrangian Relaxation

Lagrangian Dual Problem

$$\min_{\lambda \geq 0} \quad \max_{y \in Y} \quad f(y, \lambda)$$

Efficient minimization of the dual

- Max: Maximum Spanning Arborescence
- Min: Subgradient descent
- Many relaxed constraints: Non Delayed Relax-and-Cut

Efficient maximization of the primal

- Branch-and-Bound
- Problem reduction (exact pruning technique)

Experiments

Problem of existing LTAG treebanks

- Projective derivation trees only
- Derivation forest

Experiments

Problem of existing LTAG treebanks

- Projective derivation trees only
- Derivation forest

Dependency treebanks

Language	Structure of 99% of trees
English	WN + 2-BBD
German	3-BBD
Dutch	WN + 3-BBD
Spanish	WN + 2-BBD
Portuguese	WN + 3-BBD

$\Rightarrow$ just test on dependency treebanks!

Experimental setup

Weighting model

Feature-based model learned with the perceptron algorithm

Goals

- decoding time?
- accuracy?

Experimental setup

Weighting model

Feature-based model learned with the perceptron algorithm

Goals

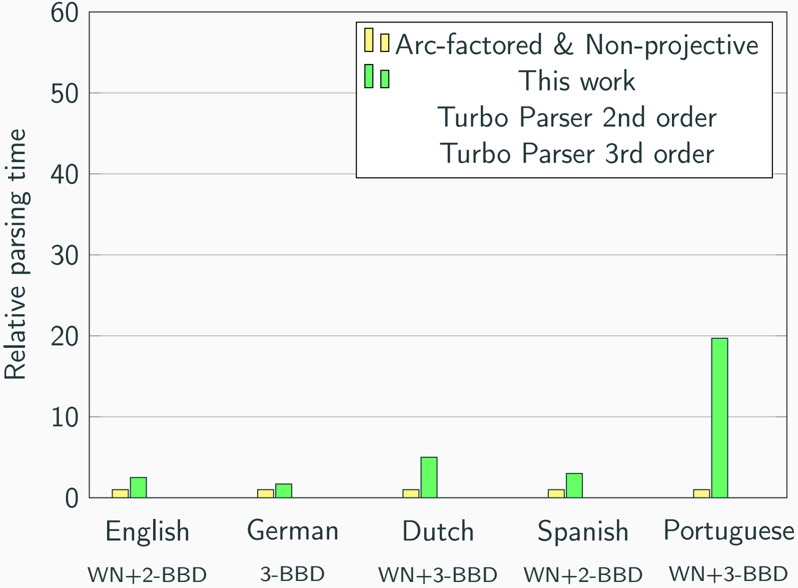
- decoding time?
- accuracy?

Turboparser [Martins et al. 2013]

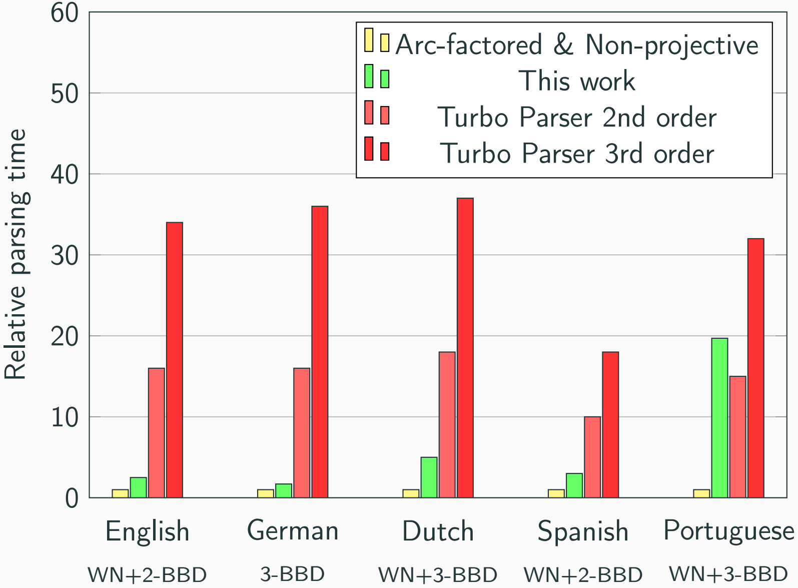
	English	German	Dutch	Spanish	Portuguese
	WN+2-BBD	3-BBD	WN+3-BBD	WN+2-BBD	WN+3-BBD
1st	94.87	98.74	93.26	93.43	94.79
2nd	99.75	99.28	97.93	98.54	98.96
3rd	99.75	99.24	97.41	99.64	98.98

Percentage of valid structure with respect to the weighting order

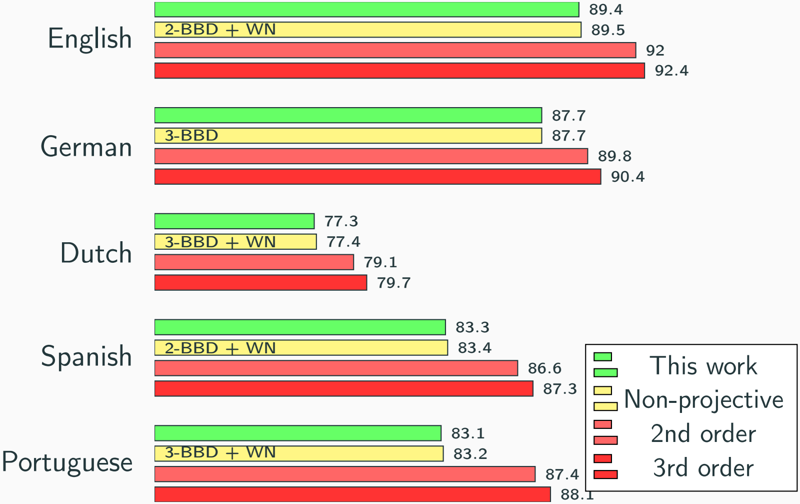
Efficiency: Relative parsing time



Efficiency: Relative parsing time



UAS (Ratio of correct arcs)



Interim conclusion

Our contribution

- First efficient and flexible algorithm:
  - k-BBD with arbitrary k
  - WN optional
- First experimental results with K-BBD and WN parsing
- Linear time algorithm LTAG parse labeller (see thesis)

Perspectives

- Applications of the algorithm to other structures (see thesis)
  $\Rightarrow$ Yield Restricted Maximum Spanning Arborescence

Limits of this approach

Pipeline issues

- Error propagation
- Possibly infeasible labelling

LTAG limits

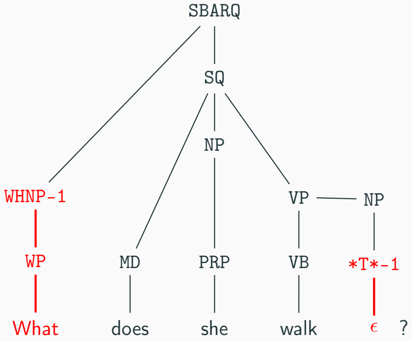
- No dataset
- Continuous constituents only

Proposal

- Joint tagging and parsing
- No LTAG motivated structural constraints

4. Joint Tagging and Dependency Parsing

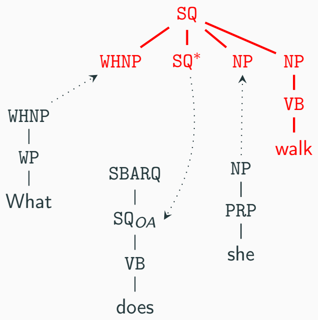
Discontinuous constituents



Motivation

- Traces usually ignored

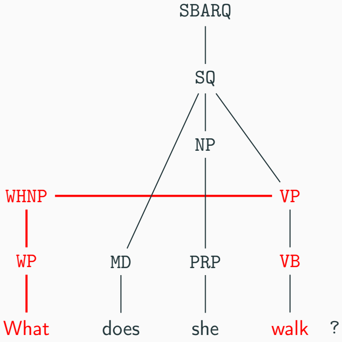
Discontinuous constituents



Motivation

- Traces usually ignored
- Difficult to automatically extract a LTAG

Discontinuous constituents



Motivation

- Traces usually ignored
- Difficult to automatically extract a LTAG
- Painless discontinuous transformation [Evang et al. 2011]

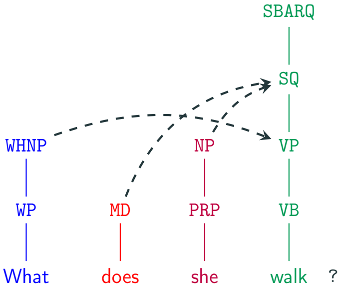
Joint tagging and dependency parsing

Problem

1. Assign one tag per lexical item
2. Assign one head per lexical item with arborescence constraints

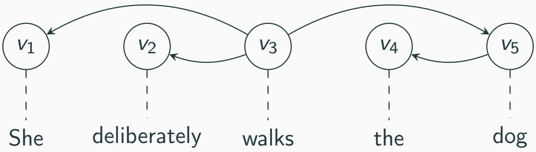
Benefits

- Flexible composition mechanism
- Guaranteed feasible solution
- More expressive weighting factors

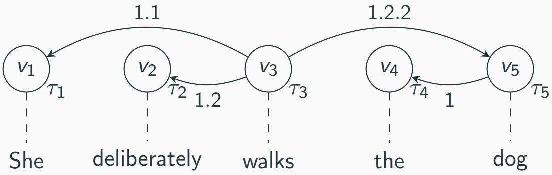


Example (1)

Example (1)

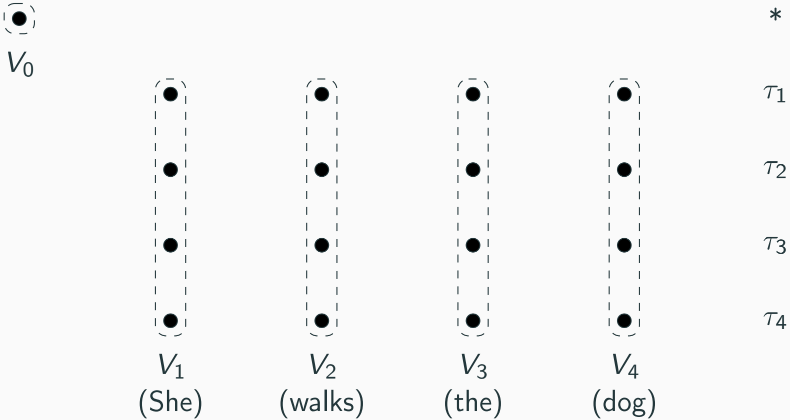


Example (1)

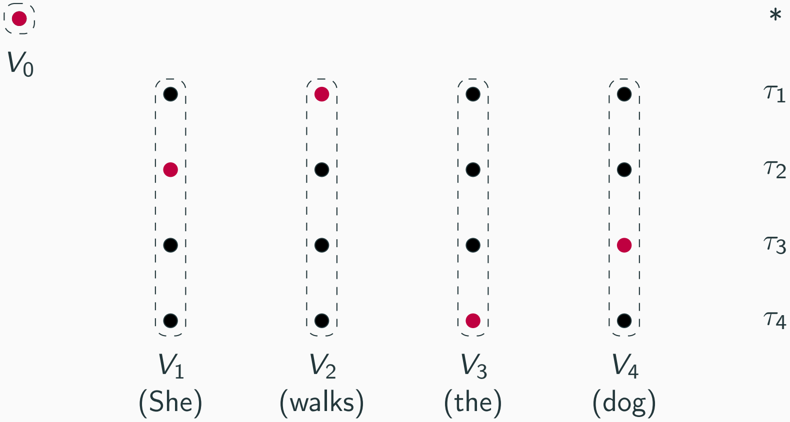


1.1
1.2.2
v_1
v_2
v_3
v_4
v_5
τ_1
τ_2
1.2
τ_3
τ_4
1
τ_5
She
deliberately
walks
the
dog

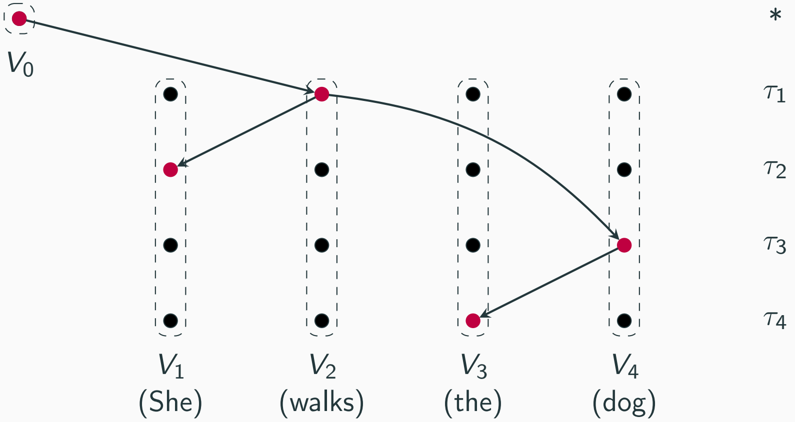
Example (2)



Example (2)



Example (2)



Generalized Maximum Spanning Arborescence (GMSA)

Reduction

- Word $\Rightarrow$ Cluster
- Tag $\Rightarrow$ vertex
- Attachment $\Rightarrow$ arc

Complexity

NP-hard [Myung et al. 1995]

Generalized Maximum Spanning Arborescence (GMSA)

Reduction

- Word ⇒ Cluster
- Tag ⇒ vertex
- Attachment ⇒ arc

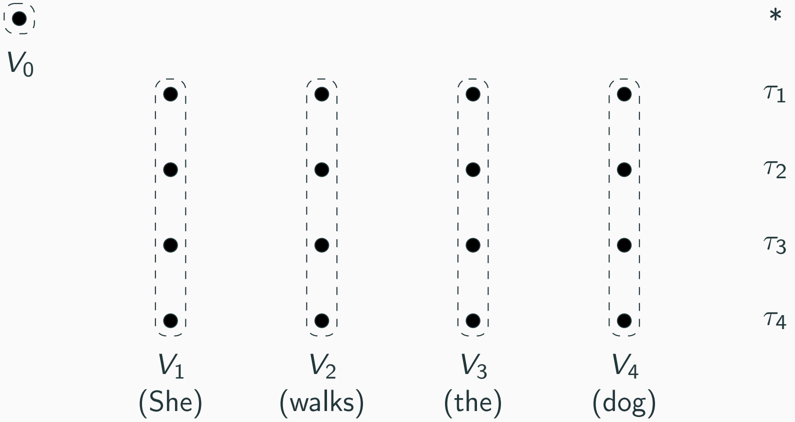
Complexity

NP-hard [Myung et al. 1995]

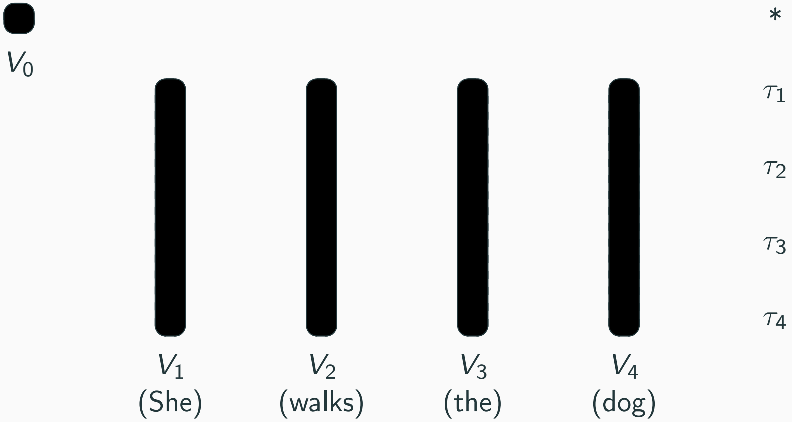
Methodology [Corro et al. 2017a]

1. Graph characterization of joint tagging and parsing
2. ILP formulation of the problem [Pop 2009]
3. Lagrangian based decoder (dual decomposition)

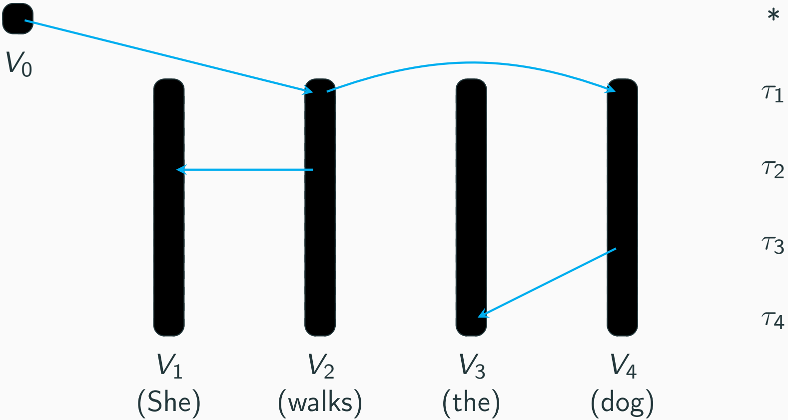
Sketch of the algorithm (1)



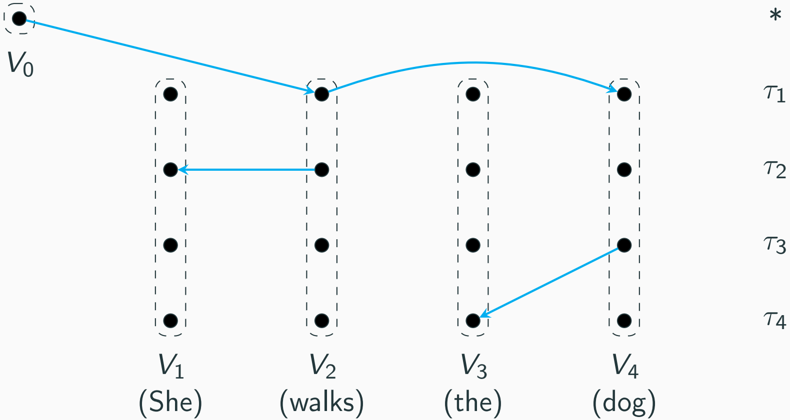
Sketch of the algorithm (1)



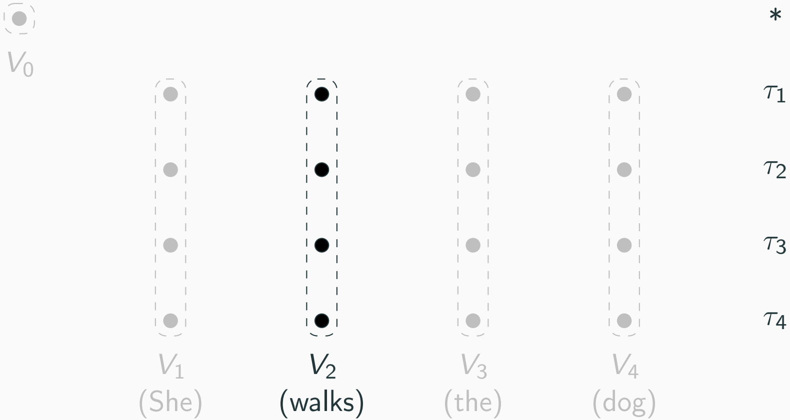
Sketch of the algorithm (1)



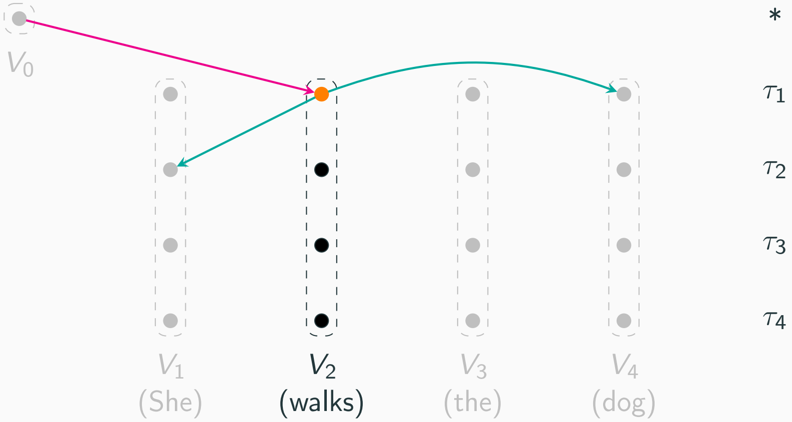
Sketch of the algorithm (1)



Sketch of the algorithm (2)



Sketch of the algorithm (2)



Dual decomposition

Lagrangian Dual Problem

$$\max_{y^1, y^2} \quad f(y^1) + \ g(y^2)$$
$$\text{s.t.} \quad y^1 = y^2$$

Dual decomposition

Lagrangian Dual Problem

$$\min_{\lambda^1, \lambda^2} \quad \max_{y^1, y^2} \quad f'(y^1, \lambda^1) + \; g'(y^2, \lambda^2)$$

Dual decomposition

Lagrangian Dual Problem

$$\min_{\lambda^1,\lambda^2} \quad \max_{y^1,y^2} \quad f'(y^1,\lambda^1) + g'(y^2,\lambda^2)$$

Efficient minimization of the dual

- Max: 2 subproblems
- Min: Subgradient descent

Lagrangian enhancement

- Arc re-weighting
- Problem reduction (exact pruning technique)

Probabilistic model

$$\begin{aligned} P(\mathbf{d}, \mathbf{t}|\mathbf{s}) &= P_\alpha(\mathbf{d}|\mathbf{s}) \times P_\nu(\mathbf{t}|\mathbf{d}, \mathbf{w}) \\ &= \prod_{(h,m)\in\mathbf{d}} P_\alpha(h|m, \mathbf{s}) \times P_\nu(t_m|m, \mathbf{d}, \mathbf{w}) \end{aligned}$$

Independence assumption

- P_α: head probability
- P_ν: tag probability conditioned on dependencies

Parameter estimation

- Neural network
- Log-likelihood maximization on train data

Experimental results

Discontinuous PTB (English)

	LF	Time (min)
	Short sentences only	
This work	**89.85**	≈ 4
van Craenburgh et al.	87.00	≈ 180
	Full test set	
This work	**89.17**	≈ 5.5

TIGER (German)

	LF	Time (min)
This work	**81.63**	≈ 11
Coavoux & Crabbé	81.60	≈ 2.5

Interim conclusion

Problem formulation

- Joint sequence tagging and non-projective dependency parsing

Contribution

- A novel approach for discontinuous constituent parsing
- A novel algorithm for the GMSA

Future work

- Max-margin training
- High-order scoring models:
  - bi-gram
  - sibling and grand-father
- Application to other joint tagging and parsing problems

5. Conclusion

Conclusion: Contributions

Methodology

1. Graph characterization of a NLP problem
2. ILP formulation
3. Lagrangian based decoder

Alternative interpretation of syntactic structures

1. LTAG derivation tree
   $\Rightarrow$ Yield Restricted Spanning Arborescence
2. Joint tagging and parsing
   $\Leftrightarrow$ Generalized Spanning Arborescence

Conclusion: Research directions

In progress

- Joint part-of-speech tagging and dependency parsing
- High-order GMSA

Lexicalized grammars [Kuhlmann 2010]

- Lexicalized LCFRS

Applications outside NLP

- Standard optimization dataset
- Other applied research areas

Conclusion: Structured latent variables

Motivation

- Syntactic parsing: (most often) not an end in itself
- Annotation process: expensive

End-to-end learning

- Syntactic structure as a layer in a neural network
- Training for the end goal (e.g. translation)

Deep generative models [Kingma et al. 2014]

- Semisupervised/unsupervised structured learning
- Linguistically motivated priors

References

Bodirsky, Manuel, Marco Kuhlmann, and Mathias Möhl (2009). "Well-nested drawings as models of syntactic structure". In: Tenth Conference on Formal Grammar and Ninth Meeting on Mathema
pp. 195–203.

Corro, Caio et al. (2016). "Dependency Parsing with Bounded Block Degree and Well-nestedness via Lagrangian Relaxation and Branch-and-Bound". In: Proceedings of the 54th Annual Meeting of the Association for Comput Berlin, Germany: Association for Computational Linguistics, pp. 355–366. DOI: 10.18653/v1/P16-1034. URL: http://www.aclweb.org/anthology/P16-1034.

Corro, Caio, Joseph Le Roux, and Mathieu Lacroix (2017a). "Efficient Discontinuous Phrase-Structure Parsing via the Generalized Maximum Spanning Arborescence". In: Proceedings of the 2017 Conference on Empirical Methods in Natural L Copenhagen, Denmark: Association for Computational Linguistics, pp. 1644–1654. URL: http://aclweb.org/anthology/D17-1172.

Corro, Caio and Joseph Le Roux (2017b). “Transforming Dependency Structures to LTAG Derivation Trees”. In: Proceedings of the 13th International Workshop on Tree Adjoining Gran
Umeå, Sweden: Association for Computational Linguistics, pp. 112–121. URL: http://aclweb.org/anthology/W17-6212.

Das, Dipanjan, André FT Martins, and Noah A Smith (2012). “An exact dual decompo-
sition algorithm for shallow semantic parsing with constraints”. In: Proceedings of the First Joint Conference on Lexical and Computationa
Association for Computational Linguistics, pp. 209–217.

Edmonds, Jack (1967). “Optimum branchings”. In: Journal of Research of the National Bureau of Standards 71.4, pp. 233–240.

Eisner, Jason (2000). "Bilexical grammars and their cubic-time parsing algorithms". In: Advances in probabilistic and other parsing technologies. Springer, pp. 29–61.

Eisner, Jason and Giorgio Satta (2000). "A faster parsing algorithm for lexicalized tree-adjoining grammars". In: Proceedings of the 5th Workshop on Tree-Adjoining Grammars and Rel pp. 14–19.

Evang, Kilian and Laura Kallmeyer (2011). "PLCFRS Parsing of English Discontinuous Constituents". In: Proceedings of the 12th International Conference on Parsing Technolog Dublin, Ireland, pp. 104–116.

Fernández-González, Daniel and André F. T. Martins (2015). "Parsing as Reduction". In: Proceedings of the 53rd Annual Meeting of the Association for Comput
Beijing, China: Association for Computational Linguistics, pp. 1523–1533. DOI: 10.3115/v1/P15-1147. URL: http://www.aclweb.org/anthology/P15-1147.

Gómez-Rodríguez, Carlos, David Weir, and John Carroll (2009). "Parsing mildly non-projective dependency structures". In: Proceedings of the 12th Conference of the European Chapter of the As
Association for Computational Linguistics, pp. 291–299.

Kingma, Diederik P et al. (2014). "Semi-supervised Learning with Deep Generative Models". In: Advances in Neural Information Processing Systems 27. Ed. by Z. Ghahramani et al. Curran Associates, Inc., pp. 3581–3589. URL: http://papers.nips.cc/paper/5352-semi-supervised-learning-with-deep-generative-models.pdf.

Kong, Lingpeng, M. Alexander Rush, and A. Noah Smith (2015). "Transforming Dependencies into Phrase Structures". In: Proceedings of the 2015 Conference of the North American Chapter of Denver, Colorado, pp. 788–798.

Koo, Terry et al. (2010). "Dual decomposition for parsing with non-projective head automata". In: Proceedings of the 2010 Conference on Empirical Methods in Natural L Association for Computational Linguistics, pp. 1288–1298.

Kuhlmann, Marco (2010). Dependency Structures and Lexicalized Grammars: An Algebraic Approa Vol. 6270. Springer.

Le Roux, Joseph, Antoine Rozenknop, and Jennifer Foster (2013). "Combining PCFG-LA models with dual decomposition: A case study with function labels and binarization". In: the 2013 Conference on Empirical Methods in Natural Language Proces

Martins, Andre, Miguel Almeida, and Noah A. Smith (2013). "Turning on the Turbo: Fast Third-Order Non-Projective Turbo Parsers". In: Proceedings of the 51st Annual Meeting of the Association for Comput Sofia, Bulgaria: Association for Computational Linguistics, pp. 617–622. URL: http://www.aclweb.org/anthology/P13-2109.

McDonald, Ryan et al. (2005). "Non-projective dependency parsing using spanning tree algorithms". In: Proceedings of the conference on Human Language Technology and Em Association for Computational Linguistics, pp. 523–530.

Myung, Young-Soo, Chang-Ho Lee, and Dong-Wan Tcha (1995). "On the generalized minimum spanning tree problem". In: Networks 26.4, pp. 231–241.

Pop, Petrica Claudiu (2009). "A survey of different integer programming formulations of the generalized minimum spanning tree problem". In: Carpathian Journal of Mathematics 25.1, pp. 104–118.

Rambow, Owen and Aravind Joshi (1997). "A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena". In: Recent trends in meaning-text theory 39, pp. 167–190.

Rush, Alexander M et al. (2010). "On Dual Decomposition and Linear Programming Relaxations for Natural Language Processing". In: Proceedings of the 2010 Conference on Empirical Methods in Natural L Cambridge, MA: Association for Computational Linguistics, pp. 1–11. URL: http://www.aclweb.org/anthology/D10-1001.

References x

Satta, Giorgio (1992). "RECOGNITION OF LINEAR CONTEXT-FREE REWRITING SYSTEMS". In: 30th Annual Meeting of the Association for Computational Linguistics. URL: http://www.aclweb.org/anthology/P92-1012.

Schrijver, A. (2003). Combinatorial Optimization - Polyhedra and Efficiency. Springer.

Zaslavskiy, Mikhail, Marc Dymetman, and Nicola Cancedda (2009). "Phrase-Based Statistical Machine Translation as a Traveling Salesman Problem". In: Proceedings of the Joint Conference of the 47th Annual Meeting of the Suntec, Singapore: Association for Computational Linguistics, pp. 333–341. URL: http://www.aclweb.org/anthology/P09-1038.