Turbo Parser Redux: From Dependencies to Constituents

André Martins

Joint work with: Noah Smith, Mário Figueiredo, Eric Xing, Pedro Aguiar, Miguel Almeida, Mariana Almeida, and Daniel Fernández-González

LxMLS, Lisboa, 26/07/16

Structured Prediction and NLP

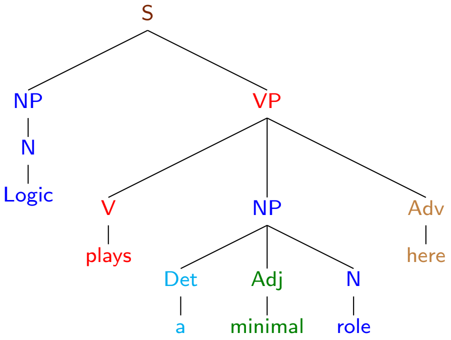
Structured prediction: a machine learning framework for predicting structured, constrained, and interdependent outputs

NLP deals with *structured* and *ambiguous* textual data:

- machine translation
- speech recognition
- syntactic parsing
- semantic parsing
- information extraction
- ...

Constituent/Phrase-Structure Parsing

```
S --> NP VP
NP --> Det Adj N
VP --> V NP Adv
Adj --> minimal
Adv --> here
Det --> a
N --> logic
N --> role
V --> plays
```

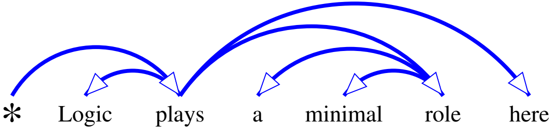


Example extracted from the Penn Treebank.

(Magerman, 1995; Charniak, 1996; Johnson, 1998; Collins, 1999; Klein and Manning, 2003)

Dependency Parsing

Map **sentences** to their **syntactic structure.**



- A lexicalized syntactic formalism
- Grammar functions represented as lexical relationships (dependencies)

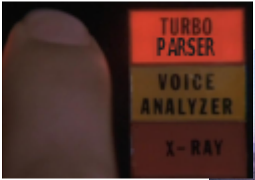
(Eisner, 1996; McDonald et al., 2005; Nivre et al., 2006; Koo et al., 2007)

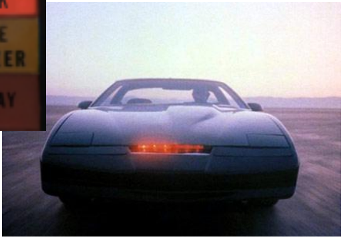
Outline

1. **Turbo Parsers**

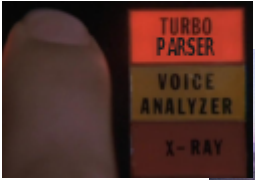
2. **Parsing as Reduction**
   - Dependencies and Constituents
   - Head-Ordered Dependency Trees
   - Reduction-Based Constituent Parsers
   - Experiments
   - Conclusions

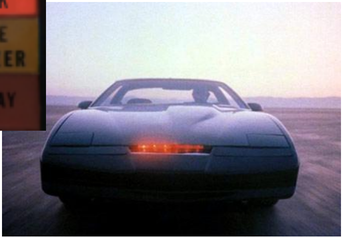
What is a Turbo Parser?





What is a Turbo Parser?





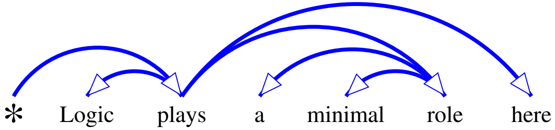
- **A parser that runs inference in factor graphs, ignoring global effects caused by loops (Martins et al., 2010)**
- name inspired from *turbo* decoders (Berrou et al., 1993)

Examples of Turbo Parsers

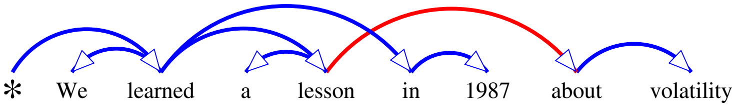
- Exponential-sized ILP formulation (Riedel and Clarke, 2006)
- Polynomial-sized ILP formulation with multi-commodity flows (Martins et al., 2009)
- Belief propagation decoder (Smith and Eisner, 2008; Martins et al., 2010)
- Dual decomposition decoder (Koo et al., 2010)
- AD^3 decoder (Martins et al., 2011, 2013)

An Important Distinction

- A projective tree:

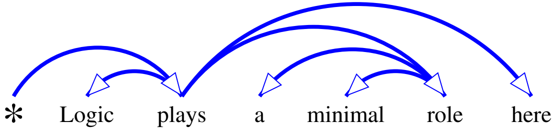


- A non-projective tree:

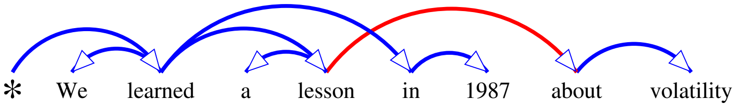


An Important Distinction

- A projective tree:



- A non-projective tree:

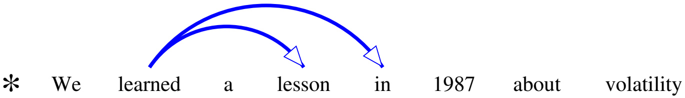


Non-projective trees are suitable for languages with flexible word order (Dutch, German, Czech,...).

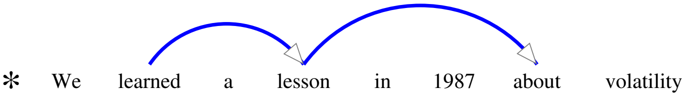
First-Order Scores for Arcs

* We learned a lesson in 1987 about volatility

Second-Order Scores for Consecutive Siblings



Second-Order Scores for Grandparents



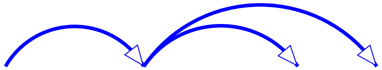
Scores for Arbitrary Siblings

* We learned a lesson in 1987 about volatility

Scores for Head Bigrams

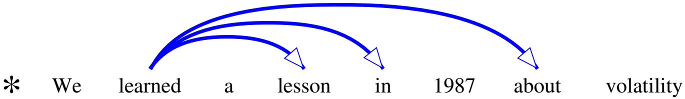
* We learned a lesson in 1987 about volatility

Third-Order Scores for Grand-siblings

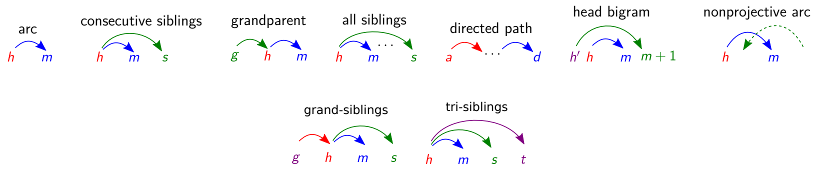


* We learned a lesson in 1987 about volatility

Third-Order Scores for Tri-siblings

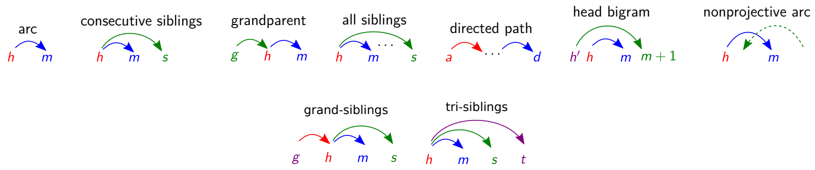


Decoding



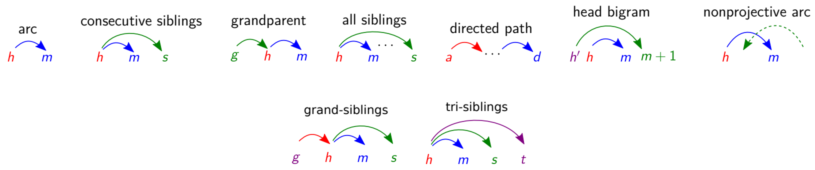
- How to deal with all these parts?

Decoding



- How to deal with all these parts?
- Beyond arc-factored models, non-projective parsing is **NP-hard** (McDonald and Satta, 2007)—**need to embrace approximations!**

Decoding



- How to deal with all these parts?
- Beyond arc-factored models, non-projective parsing is **NP-hard** (McDonald and Satta, 2007)—**need to embrace approximations!**

	parser	AF	CS	G	AS	DP	HB	NPA	GS	TS
McDonald et al. (2006)	projective + greedy	✓	✓							
Smith et al. (2008)	loopy BP	✓	✓	✓	✓					
Martins et al. (2010)	LP solver	✓		✓	✓			✓		
Koo et al. (2010)	dual decomp.	✓	✓							
Martins et al. (2011)	AD^3	✓	✓	✓	✓	✓	✓	✓		
Martins et al. (2013)	AD^3 & active set	✓	✓	✓	✓		✓		✓	✓

Factor Graph Representations

- For each input $x \in \mathcal{X}$: a **large** set of candidate outputs $\mathcal{Y}(x)$
- **Decoding problem:**

$$\widehat{y} = \arg \max_{y \in \mathcal{Y}(x)} F_{\mathbf{w}}(x, y)$$

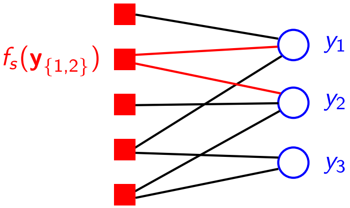
Factor Graph Representations

- For each input $x \in \mathcal{X}$: a **large** set of candidate outputs $\mathcal{Y}(x)$
- **Decoding problem:**

$$\widehat{y} = \arg \max_{y \in \mathcal{Y}(x)} F_{\mathbf{w}}(x, y)$$

- **Key assumption:** $F_{\mathbf{w}}$ decomposes into (overlapping) *parts*

$$\boxed{F_{\mathbf{w}}(x, y) := \sum_{s} f_s(\mathbf{y}_s)}$$



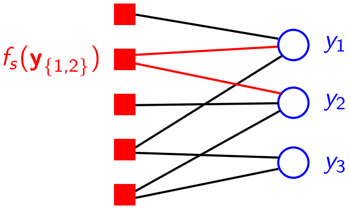
Factor Graph Representations

- For each input $x \in \mathcal{X}$: a **large** set of candidate outputs $\mathcal{Y}(x)$
- **Decoding problem:**

$$\widehat{y} = \arg\max_{y \in \mathcal{Y}(x)} F_{\mathbf{w}}(x, y)$$

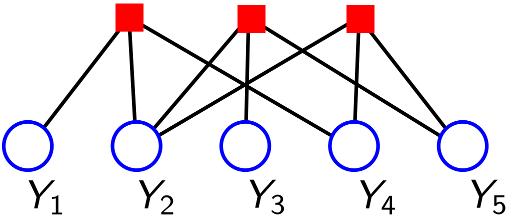
- **Key assumption:** $F_{\mathbf{w}}$ decomposes into (overlapping) *parts*

$$\boxed{F_{\mathbf{w}}(x, y) := \sum_{s} f_s(\mathbf{y}_s)}$$

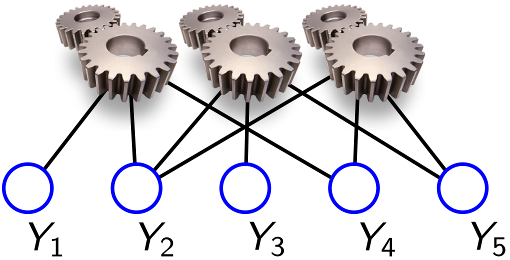


- **Examples:** HMMs, CRFs, PCFGs, general graphical models

Factors as Machines



Factors as Machines



Alternating Directions Dual Decomposition (AD^3)

A general purpose algorithm, suitable for many scenarios in NLP and IR.

Alternating Directions Dual Decomposition (AD^3)

A general purpose algorithm, suitable for many scenarios in NLP and IR.

- Problems with factor graph representations

Alternating Directions Dual Decomposition (AD^3)

A general purpose algorithm, suitable for many scenarios in NLP and IR.

- Problems with factor graph representations
- Statements in FOL

Alternating Directions Dual Decomposition (AD^3)

A general purpose algorithm, suitable for many scenarios in NLP and IR.

- Problems with factor graph representations
- Statements in FOL
- Budget/knapsack constraints

Alternating Directions Dual Decomposition (AD^3)

A general purpose algorithm, suitable for many scenarios in NLP and IR.

- Problems with factor graph representations
- Statements in FOL
- Budget/knapsack constraints
- Combination of structured models

Alternating Directions Dual Decomposition (AD^3)

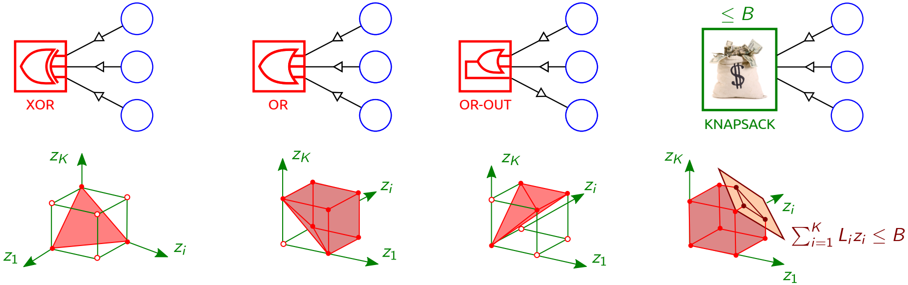
A general purpose algorithm, suitable for many scenarios in NLP and IR.

- Problems with factor graph representations
- Statements in FOL
- Budget/knapsack constraints
- Combination of structured models

High level idea:

- Decompose a complex problem into local subproblems (factors), constrained to be globally consistent
- Iterate between solving the local subproblems and penalizing the global disagreements (via Lagrange multipliers)
- FOL/knapsack constraints: the local subproblems correspond to projections onto "hard constraint" polytopes

Projecting onto Hard Constraint Polytopes



- All projections can be computed in linear time (Martins et al., 2015)
- **Applications:** Markov logic networks (Richardson and Domingos, 2006), constrained conditional models (Roth and Yih, 2004), summarization (Almeida and Martins, 2013), ...

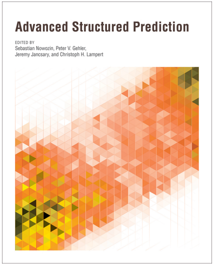
Some Problems in Which AD^3 Have Been Applied

- Dependency parsing (Martins et al., 2011, 2013)
- Frame semantics (Das et al., 2012)
- Broad-coverage semantic parsing (Martins and Almeida, 2014)
- Compressive summarization (Almeida and Martins, 2013)
- Coreference resolution (Almeida et al., 2014)

Could be a great fit to many other applications!!

Literature Pointers

- André F. T. Martins.
  "AD^3: A Fast Decoder for Structured Prediction."
  Book chapter of *Advanced Structured Prediction*, Sebastian Nowozin, Peter V. Gehler, Jeremy Jancsary, and Christoph H. Lampert (Editors), MIT Press, 2014.



- A. Martins, M. Figueiredo, P. Aguiar, N. Smith, E. Xing.
  "AD3: Alternating Directions Dual Decomposition for MAP Inference in Graphical Models."
  JMLR 2015.

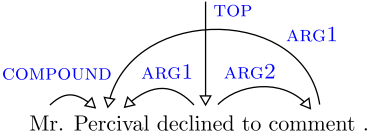
More details: EMNLP 2014 tutorial on "LP Decoders for NLP."

Parsing Accuracies/Runtimes

SOTA accuracies for the largest non-projective datasets (CoNLL-2006 and CoNLL-2008):

Extension: Broad-Coverage Semantic Parsing

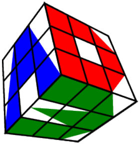
Same idea applied to **semantic role labeling**.



Best results in the SemEval 2014 shared task:

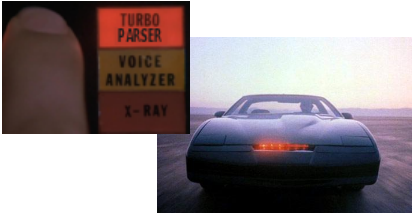
- André F. T. Martins and Mariana S. C. Almeida.
  "Priberam: A Turbo Semantic Parser with Second Order Features."
  SemEval 2014.

Try It Yourself: AD^3 Toolkit



- Freely available at: *http://www.ark.cs.cmu.edu/AD3*
- Implemented in C++, includes a Python wrapper (thanks to Andy Mueller)
- Many built-in factors: logic, knapsack, dense, and some structured factors
- You can implement your own factor (only need to write a local MAP decoder!)
- Toy examples included (parsing, coreference, Potts models)

Try It Yourself: TurboParser



- Freely available at: *http://www.ark.cs.cmu.edu/TurboParser*
- Implemented in C++, includes a Python wrapper
- Not just parsing, but a full NLP pipeline now!
- Includes multilingual POS tagging, dependency parsing, semantic role labeling, entity recognition, coreference resolution (all trainable on any dataset).

Outline

1. Turbo Parsers

2. **Parsing as Reduction**
   - Dependencies and Constituents
   - Head-Ordered Dependency Trees
   - Reduction-Based Constituent Parsers
   - Experiments
   - Conclusions

In a Nutshell (Fernández-González and Martins, 2015, ACL)

- Constituent parsers are slow (heavy grammar constant)
- Dependency parsers are faster, but their output is less informative
- How to get the best of both worlds?

In a Nutshell (Fernández-González and Martins, 2015, ACL)

- Constituent parsers are slow (heavy grammar constant)
- Dependency parsers are faster, but their output is less informative
- How to get the best of both worlds?

Our proposal: a reduction of constituent parsing to dependency parsing

In a Nutshell (Fernández-González and Martins, 2015, ACL)

- Constituent parsers are slow (heavy grammar constant)
- Dependency parsers are faster, but their output is less informative
- How to get the best of both worlds?

Our proposal: a reduction of constituent parsing to dependency parsing

- Rooted in a novel formalism: **head-ordered dependency trees**
- Works for **any out-of-the-box dependency parser**
- Competitive for English and morphologically rich languages
- Results above the state of the art for **discontinuous parsing**

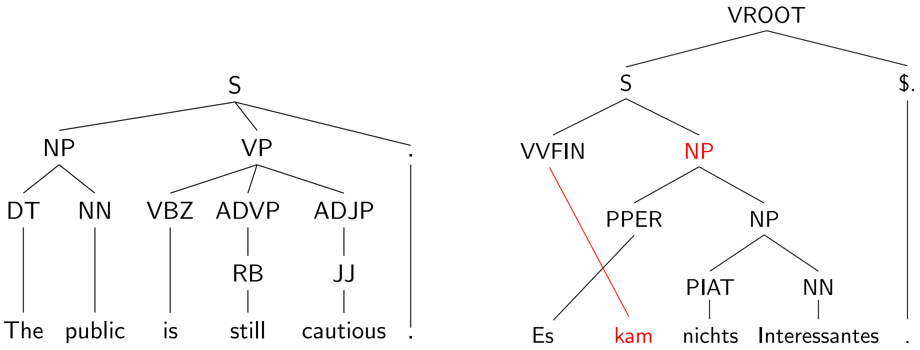
Outline

1. **Turbo Parsers**

2. **Parsing as Reduction**
   - Dependencies and Constituents
   - Head-Ordered Dependency Trees
   - Reduction-Based Constituent Parsers
   - Experiments
   - Conclusions

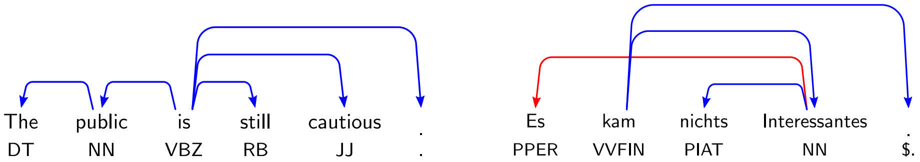
Outline

Continuous and Discontinuous C-Trees



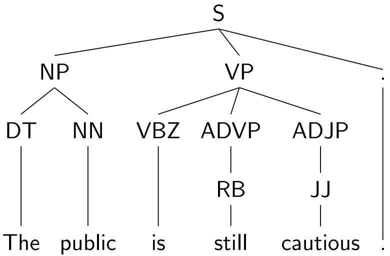
- CFG generate **continuous** trees, LCFRS generate **discontinuous** trees (Vijay-Shanker et al., 1987)
- ... but existing discontinuous parsers are too slow and inaccurate!

Projective and Non-Projective D-Trees

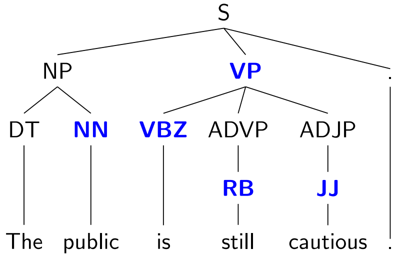


- Continuous and discontinuous c-trees “project” respectively to **projective** and **non-projective** d-trees (Gaifman, 1965)
- Non-projectiveness is suitable for languages with flexible word order (Dutch, German, Czech, etc.)

Projecting C-Trees onto D-Trees...



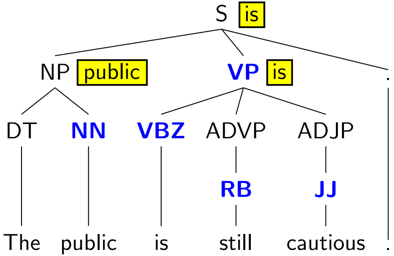
Projecting C-Trees onto D-Trees...



1. apply set of head rules:

S	$\rightarrow$	NP VP .
NP	$\rightarrow$	DT NN
VP	$\rightarrow$	VBZ ADVP ADJP
		...

Projecting C-Trees onto D-Trees...

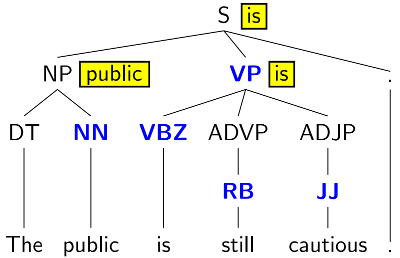


1. apply set of head rules:

   S → NP **VP** .
   NP → DT **NN**
   VP → **VBZ** ADVP ADJP
   ...

2. lexicalize

Projecting C-Trees onto D-Trees...

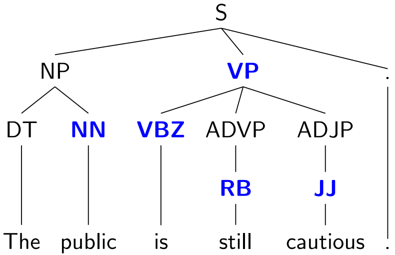


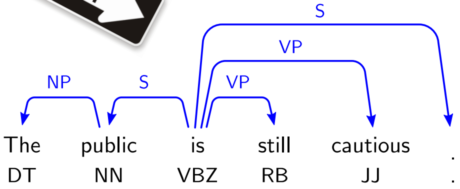
1. apply set of head rules:

S	$\rightarrow$	NP VP .
NP	$\rightarrow$	DT NN
VP	$\rightarrow$	VBZ ADVP ADJP
		...

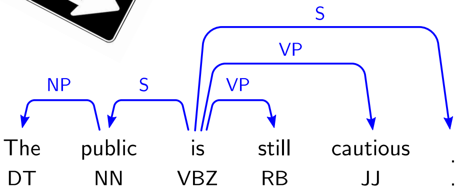
2. lexicalize
3. drop constituent nodes

Projecting C-Trees onto D-Trees...





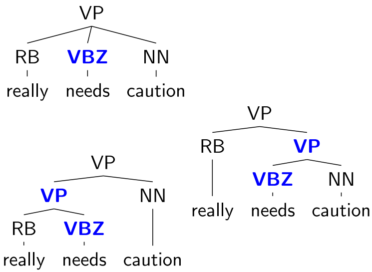
... And Back?

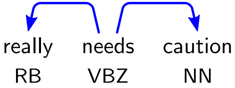


… And Back?

really	needs	caution
RB	VBZ	NN

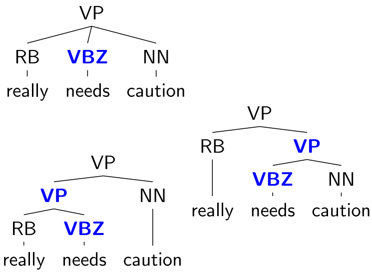
… And Back?

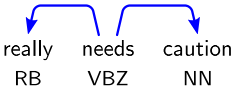




left-branch? right-branch? flat?

… And Back?





left-branch? right-branch? flat?

This paper: formal equivalence results to "invert" this projection.

Related Work

- Store structural information in the dependency labels (Hall and Nivre, 2008)
- Manual transformation rules toward multi-representational treebanks (Xia and Palmer, 2001; De Marneffe et al., 2006; Xia et al., 2008)
- Apply second-stage constituent parser (Kong et al., 2015)
- Joint dependency and constituent parsing (Carreras et al., 2008; Rush et al., 2010)

Outline

1. Turbo Parsers

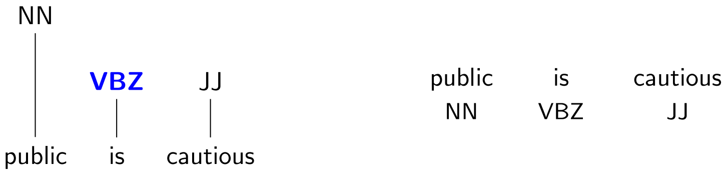
2. **Parsing as Reduction**
   - Dependencies and Constituents
   - Head-Ordered Dependency Trees
   - Reduction-Based Constituent Parsers
   - Experiments
   - Conclusions

Strictly Ordered D-Trees

Key idea: endow d-trees with additional structure, by making each head attach its modifiers in a particular order

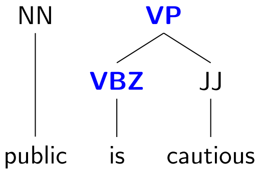
Strictly Ordered D-Trees

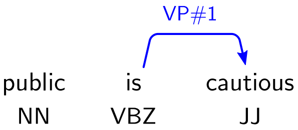
Key idea: endow d-trees with additional structure, by making each head attach its modifiers in a particular order



Strictly Ordered D-Trees

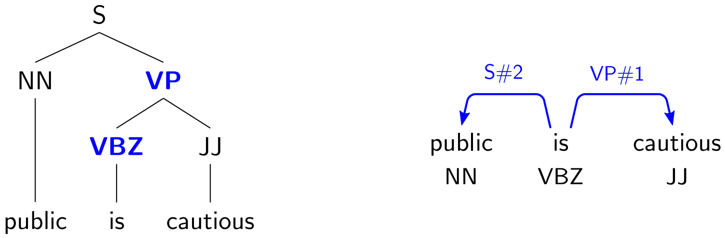
Key idea: endow d-trees with additional structure, by making each head attach its modifiers in a particular order





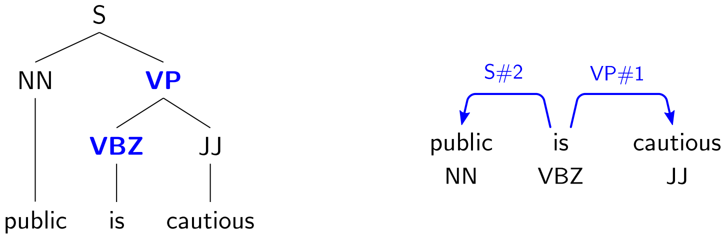
Strictly Ordered D-Trees

Key idea: endow d-trees with additional structure, by making each head attach its modifiers in a particular order



Strictly Ordered D-Trees

Key idea: endow d-trees with additional structure, by making each head attach its modifiers in a particular order

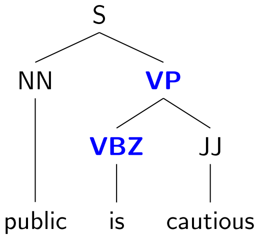


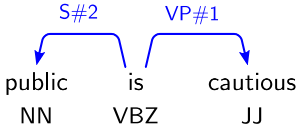
Proposition

Binary c-trees = strictly ordered d-trees

Strictly Ordered D-Trees

Key idea: endow d-trees with additional structure, by making each head attach its modifiers in a particular order





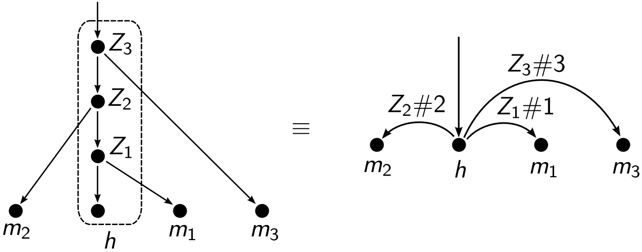
Proposition

Binary c-trees = strictly ordered d-trees

- Same number of symbols (dependency alphabet = phrasal alphabet)

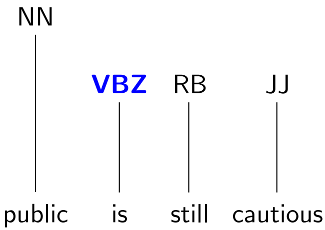
The Spinal View

- The order is given by the attachment position in the **spine** (Carreras et al., 2008)



Weakly Ordered D-Trees

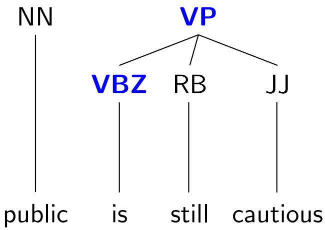
Same, but allow **simultaneous** events (as long as the d-label is consistent)

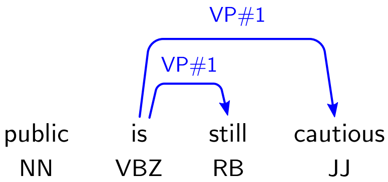


public	is	still	cautious
NN	VBZ	RB	JJ

Weakly Ordered D-Trees

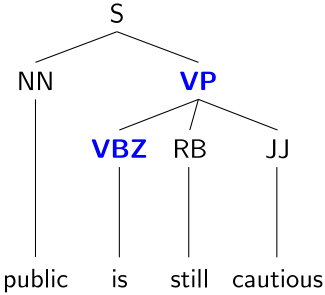
Same, but allow **simultaneous** events (as long as the d-label is consistent)

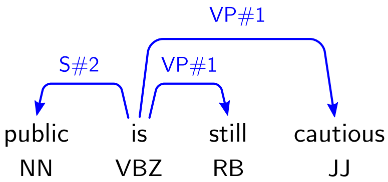




Weakly Ordered D-Trees

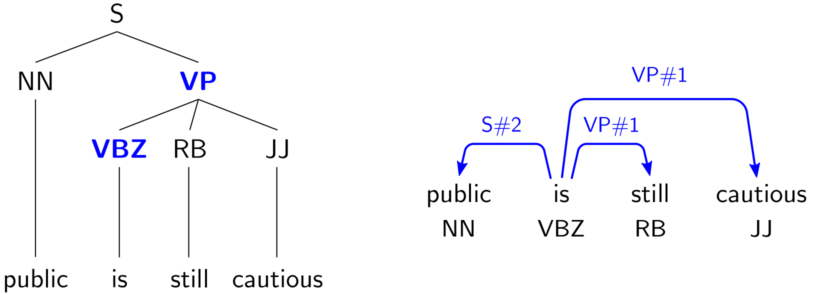
Same, but allow **simultaneous** events (as long as the d-label is consistent)





Weakly Ordered D-Trees

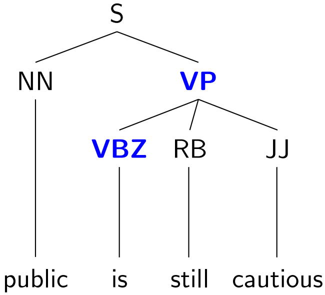
Same, but allow **simultaneous** events (as long as the d-label is consistent)

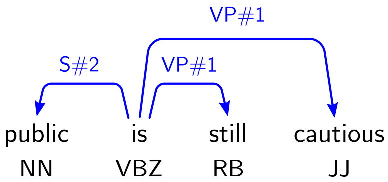


- Can every c-tree be represented like this?

Weakly Ordered D-Trees

Same, but allow **simultaneous** events (as long as the d-label is consistent)

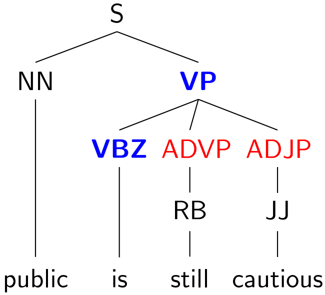


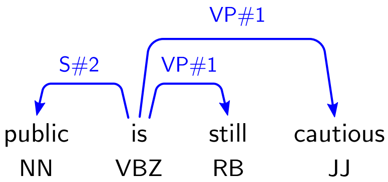


- Can every c-tree be represented like this? **No: unaries are lost.**

Weakly Ordered D-Trees

Same, but allow **simultaneous** events (as long as the d-label is consistent)

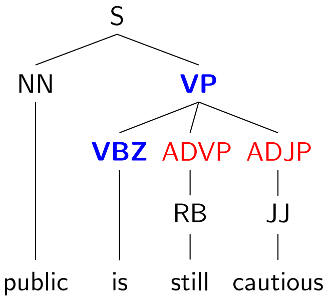


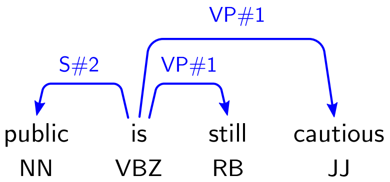


- Can every c-tree be represented like this? **No: unaries are lost.**

Weakly Ordered D-Trees

Same, but allow **simultaneous** events (as long as the d-label is consistent)





- Can every c-tree be represented like this? **No: unaries are lost.**

Proposition

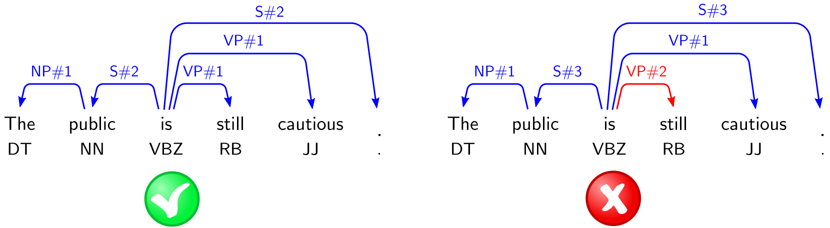
Unaryless c-trees = weakly ordered d-trees

What About Projective Trees?

A head-ordered d-tree has the **nesting property** if, on each side of every head, closer modifiers are attached first.

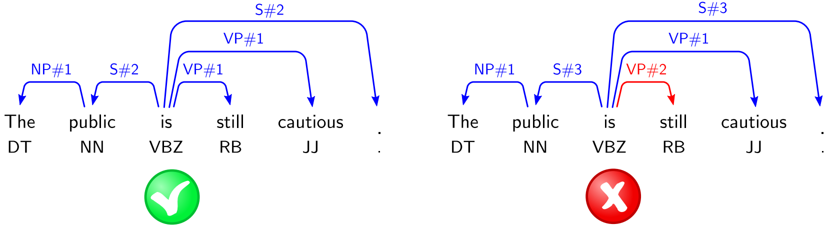
What About Projective Trees?

A head-ordered d-tree has the **nesting property** if, on each side of every head, closer modifiers are attached first.



What About Projective Trees?

A head-ordered d-tree has the **nesting property** if, on each side of every head, closer modifiers are attached first.

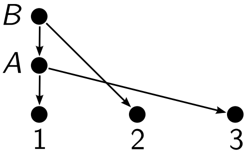


Proposition

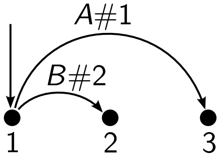
Unaryless continuous c-trees = nested-weakly ordered projective d-trees

The Spinal View for Discontinuities

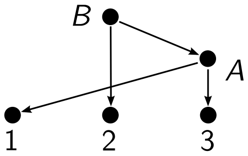
- Projective, but not nested:



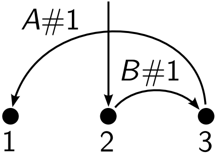
$\equiv$



- Nested, but not projective:



$\equiv$



Outline

Reduction-Based Constituent Parsers

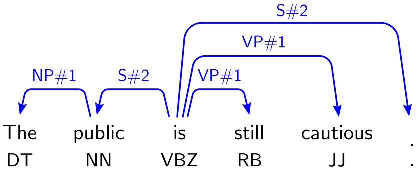
1. Convert c-treebank to head-ordered d-treebank. ✓
2. Encode head-orders in the d-labels, yielding a d-treebank.
3. Train a d-parser on the d-treebank.
4. Run the d-parser on new sentences. ✓
5. Convert the predicted d-trees into unaryless c-trees. ✓
6. Recover unary nodes.

Reduction-Based Constituent Parsers

1. Convert c-treebank to head-ordered d-treebank. ✓
2. Encode head-orders in the d-labels, yielding a d-treebank.
3. Train a d-parser on the d-treebank.
4. Run the d-parser on new sentences. ✓
5. Convert the predicted d-trees into unaryless c-trees. ✓
6. Recover unary nodes.

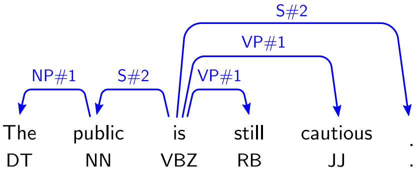
Label Encoding Strategies

direct encoding

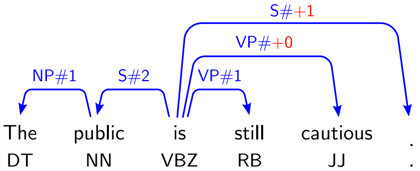


Label Encoding Strategies

direct encoding

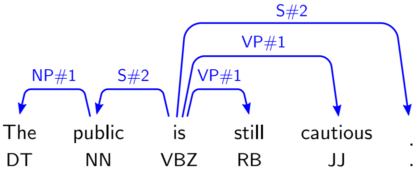


delta encoding

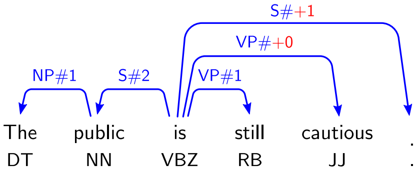


Label Encoding Strategies

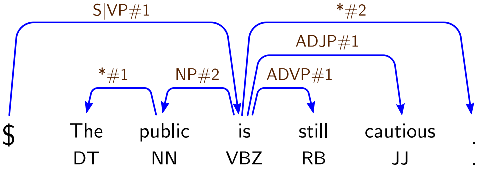
direct encoding



delta encoding



H&N encoding (Hall and Nivre, 2008)



Impact of Label Encoding

- Evaluated on the English PTB §22 (Marcus et al., 1993).

	# labels	dep (LAS)	const (F_1)
H&N encoding	731	87.86	89.39
Direct encoding	75	91.99	90.89
Delta encoding	**69**	**92.00**	**90.94**

- H&N encoding overgenerates labels, leading to a loss in accuracy
- Delta encoding performs consistently better than direct encoding on other datasets (see paper)

Reduction-Based Constituent Parsers

1. Convert c-treebank to head-ordered d-treebank. ✓
2. Encode head-orders in the d-labels, yielding a d-treebank. ✓
3. Train a d-parser on the d-treebank.
4. Run the d-parser on new sentences. ✓
5. Convert the predicted d-trees into unaryless c-trees. ✓
6. Recover unary nodes.

Choice of Dependency Parser

- Evaluated on the English PTB §22 (Marcus et al., 1993).

Dependency Parser	**Dep (LAS)**	**Const (F_1)**	**# toks/s.**
MaltParser	88.95	86.87	5,392
MSTParser	89.86	87.93	363
ZPar	91.28	89.50	1,022
TurboParser-Basic	90.23	87.63	2,585
TurboParser-Standard	91.58	90.41	1,658
TurboParser-Full	91.70	90.53	959
TurboParser-Full + Labeler	**92.00**	**90.94**	912

- Best results: separate stages for d-parser and d-labeler
- The d-labeler is a simple sequence model for each head (see paper)

Reduction-Based Constituent Parsers

1. Convert c-treebank to head-ordered d-treebank. ✓
2. Encode head-orders in the d-labels, yielding a d-treebank. ✓
3. Train a d-parser on the d-treebank. ✓
4. Run the d-parser on new sentences. ✓
5. Convert the predicted d-trees into unaryless c-trees. ✓
6. Recover unary nodes.

Recovery of Unary Nodes

- We run independent classifiers at each c-node
- Each class is either `NULL` (no unary node pre-appended) or a concatenation of labels (*e.g.*, `S->ADJP` for a node `JJ`)
- To speed-up: only observed classes are considered (9.9 classes per node in PTB §22)
- A tiny fraction of the time is spent on this post-processing ($<$2%), with F_1-score of 99.43% in PTB §22

Outline

Experiments: English PTB

- Results on the English PTB §23 (Marcus et al., 1993).

Parser	LR	LP	F1	#Toks/s.
Klein and Manning (2003)	85.3	86.5	85.9	143
Hall et al. (2014)	88.4	88.8	88.6	12
Socher et al. (2013)	*89.1*	*89.7*	*89.4*	*70*
Charniak (2000)	89.5	89.9	89.5	–
Stanford Shift-Reduce (2014)	89.1	89.1	89.1	655
Petrov and Klein (2007)	90.0	90.3	90.1	169
This work	89.9	90.4	90.2	957
Zhu et al. (2013)	90.3	90.6	90.4	1,290
Carreras et al. (2008)	90.7	91.4	91.1	–
Zhu et al. (2013)	*91.1*	*91.5*	*91.3*	–
Charniak and Johnson (2005)	*91.2*	*91.8*	*91.5*	*84*

Grayed parsers are ensemble/reranking/semi-supervised systems.

Experiments: Morphologically Rich Languages

- Results on SPMRL14 shared task datasets (Seddah et al., 2014).

Parser	Bas	Fre	Ger	Heb	Hun	Kor	Pol	Swe	Avg.
Berkeley	70.50	**80.38**	78.30	86.96	81.62	71.42	79.23	79.19	78.45
Berkeley Tagged	74.74	79.76	78.28	85.42	85.22	78.56	86.75	80.64	81.17
Crabbé and Seddah (2014)	85.35	79.68	77.15	86.19	87.51	79.35	**91.60**	82.72	83.69
Hall et al. (2014)	83.39	79.70	78.43	87.18	**88.25**	**80.18**	90.66	82.00	83.72
This work	**85.90**	78.75	**78.66**	**88.97**	88.16	79.28	91.20	**82.80**	**84.22**
Björkelund et al. (2014)	*88.24*	*82.53*	*81.66*	*89.80*	*91.72*	*83.81*	*90.50*	*85.50*	*86.72*

Experiments: Discontinuous Parsing

- Results on the discontinuous TIGER treebank (Brants et al., 2002).

	TIGER-SPMRL, $L \leq 70$	F_1	EX
gold tags	Versley (2014b)	76.46	41.05
	This work	**80.98**	**43.44**
pred. tags	Versley (2014b)	73.90	37.00
	This work	**77.72**	**38.75**

	TIGER-H&N, $L \leq 40$	F_1	EX
gold tags	Hall and Nivre (2008)	79.93	37.78
	Versley (2014a)	74.23	37.32
	This work	**85.53**	**51.21**
pred. tags	Hall and Nivre (2008)	75.33	32.63
	van Cranenburgh and Bod (2013)	78.8–	40.8–
	This work	**82.57**	**45.93**

Experiments: Discontinuous Parsing

- Results on the discontinuous NEGRA treebank (Skut et al., 1997).

	NEGRA, $L \leq 40$	**F_1**	**EX**
gold tags	van Cranenburgh (2012)	72.33	33.16
	van Cranenburgh and Bod (2013)	76.8–	40.5–
	This work	**81.08**	**48.04**
pred. tags	van Cranenburgh and Bod (2013)	74.8–	38.7–
	This work	**77.93**	**44.83**

- We parse all sentences (regardless of length) in 27.1 seconds in a single core (618 toks/sec)
- Orders of magnitude faster than van Cranenburgh and Bod (2013)
- Similar speed as the easy-first system of Versley (2014a), but much higher accuracy

Outline

1. Turbo Parsers

2. **Parsing as Reduction**
   - Dependencies and Constituents
   - Head-Ordered Dependency Trees
   - Reduction-Based Constituent Parsers
   - Experiments
   - Conclusions

Conclusions

- We proposed a **reduction technique** that allows to implement a constituent parser when only a dependency parser is available.
- The technique is very **simple** and **flexible**: applicable to any dependency parser, regardless of its nature or kind.
- If the dependency parser is non-projective, we can predict **discontinuous constituent trees**.
- We showed empirically that the reduction leads to highly-competitive constituent parsers for English and 8 morphologically rich languages.
- We surpassed the state of the art in discontinuous parsing of German by a wide margin.

We're Hiring!

Excited about MT, crowdsourcing and Lisbon? $\Rightarrow$ jobs@unbabel.com.

Unbabel

Acknowledgments

- Spanish Ministry of Economy and Competitiveness and FEDER (project TIN2010-18552-C03-01)
- Ministry of Education (FPU Grant Program) and Xunta de Galicia (projects R2014/029 and R2014/034)
- Fundação para a Ciência e Tecnologia, grants UID/EEA/50008/2013 and PTDC/EEI-SII/2312/2012.
- Priberam: QREN/POR Lisboa (Portugal), EU/FEDER programme, Intelligo project, contract 2012/24803.

References I

Almeida, M. B. and Martins, A. F. T. (2013). Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

Almeida, M. S. C., Almeida, M. B., and Martins, A. F. T. (2014). A joint model for quotation attribution and coreference resolution. In *Proc. of the Annual Meeting of the European Chapter of the Association for Computational Linguistics*.

Berrou, C., Glavieux, A., and Thitimajshima, P. (1993). Near Shannon limit error-correcting coding and decoding. In *Proc. of International Conference on Communications*, volume 93, pages 1064–1070.

Björkelund, A., Çetinoğlu, O., Faleńska, A., Farkas, R., Mueller, T., Seeker, W., and Szántó, Z. (2014). Introducing the ims-wrocław-szeged-cis entry at the spmrl 2014 shared task: Reranking and morpho-syntax meet unlabeled data. In *Proc. of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*.

Brants, S., Dipper, S., Hansen, S., Lezius, W., and Smith, G. (2002). The TIGER treebank. In *Proc. of the workshop on treebanks and linguistic theories*.

Carreras, X., Collins, M., and Koo, T. (2008). TAG, Dynamic Programming, and the Perceptron for Efficient, Feature-rich Parsing. In *Proc. of the International Conference on Natural Language Learning*.

Charniak, E. (1996). Tree-bank grammars. In *Proc. of the National Conference on Artificial Intelligence*.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proc. of the North American Chapter of the Association for Computational Linguistics Conference*.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of Annual Meeting of the Association for Computational Linguistics*.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania.

Crabbé, B. and Seddah, D. (2014). Multilingual discriminative shift reduce phrase structure parsing for the SPMRL 2014 shared task. In *Proc. of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*.

Das, D., Martins, A. F. T., and Smith, N. A. (2012). An Exact Dual Decomposition Algorithm for Shallow Semantic Parsing with Constraints. In *Proc. of First Joint Conference on Lexical and Computational Semantics (*SEM)*.

De Marneffe, M.-C., MacCartney, B., Manning, C. D., et al. (2006). Generating typed dependency parses from phrase structure parses. In *Proc. of the Meeting of the Language Resources and Evaluation Conference*.

References II

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proc. of International Conference on Computational Linguistics*.

Fernández-González, D. and Martins, A. F. T. (2015). Parsing as reduction. In *Proc. of the Conference of the Association for Computational Linguistics*.

Gaifman, H. (1965). Dependency systems and phrase-structure systems. *Information and control*.

Hall, D., Durrett, G., and Klein, D. (2014). Less grammar, more features. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

Hall, J. and Nivre, J. (2008). A dependency-driven parser for german dependency and constituency representations. In *Proc. of the Workshop on Parsing German*.

Johnson, M. (1998). PCFG models of linguistic tree representations. *Computational Linguistics*.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proc. of Annual Meeting on Association for Computational Linguistics*.

Kong, L., Rush, A. M., and Smith, N. A. (2015). Transforming dependencies into phrase structures. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Koo, T., Globerson, A., Carreras, X., and Collins, M. (2007). Structured prediction models via the matrix-tree theorem. In *Empirical Methods for Natural Language Processing*.

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *Proc. of Empirical Methods for Natural Language Processing*.

Magerman, D. (1995). Statistical decision-tree models for parsing. In *Proc. of Annual Meeting on Association for Computational Linguistics*, pages 276–283.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*.

Martins, A. F. T., Almeida, M. B., and Smith, N. A. (2013). Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

Martins, A. F. T. and Almeida, M. S. C. (2014). Priberam: A turbo semantic parser with second order features. In *Proc. of the International Workshop on Semantic Evaluations (SemEval); task 8: Broad-Coverage Semantic Dependency Parsing*.

References III

Martins, A. F. T., Figueiredo, M. A. T., Aguiar, P. M. Q., Smith, N. A., and Xing, E. P. (2015). AD^3: Alternating Directions Dual Decomposition for MAP Inference in Graphical Models. *Journal of Machine Learning Research (to appear)*.

Martins, A. F. T., Smith, N. A., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2011). Dual Decomposition with Many Overlapping Components. In *Proc. of Empirical Methods for Natural Language Processing*.

Martins, A. F. T., Smith, N. A., and Xing, E. P. (2009). Concise Integer Linear Programming Formulations for Dependency Parsing. In *Proc. of Annual Meeting of the Association for Computational Linguistics*.

Martins, A. F. T., Smith, N. A., Xing, E. P., Figueiredo, M. A. T., and Aguiar, P. M. Q. (2010). Turbo Parsers: Dependency Parsing by Approximate Variational Inference. In *Proc. of Empirical Methods for Natural Language Processing*.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proc. of Empirical Methods for Natural Language Processing*.

McDonald, R. and Satta, G. (2007). On the complexity of non-projective data-driven dependency parsing. In *Proc. of International Conference on Parsing Technologies*.

Nivre, J., Hall, J., Nilsson, J., Eryiğit, G., and Marinov, S. (2006). Labeled pseudo-projective dependency parsing with support vector machines. In *Proc. of International Conference on Natural Language Learning*.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Proc. of the North American Chapter of the Association for Computational Linguistics*.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1):107–136.

Riedel, S. and Clarke, J. (2006). Incremental integer linear programming for non-projective dependency parsing. In *Proc. of Empirical Methods for Natural Language Processing*.

Roth, D. and Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. In *International Conference on Natural Language Learning*.

Rush, A., Sontag, D., Collins, M., and Jaakkola, T. (2010). On dual decomposition and linear programming relaxations for natural language processing. In *Proc. of Empirical Methods for Natural Language Processing*.

Seddah, D., Kübler, S., and Tsarfaty, R. (2014). Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In *Proc. of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*.

References IV

Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1997). An annotation scheme for free word order languages. In *Proc. of the Fifth Conference on Applied Natural Language Processing ANLP-97*.

Smith, D. and Eisner, J. (2008). Dependency parsing by belief propagation. In *Proc. of Empirical Methods for Natural Language Processing*.

Socher, R., Bauer, J., Manning, C. D., and Ng, A. Y. (2013). Parsing with compositional vector grammars. In *Proc. of Annual Meeting of the Association for Computational Linguistics*.

van Cranenburgh, A. (2012). Efficient parsing with linear context-free rewriting systems. In *Proc. of the Conference of the European Chapter of the Association for Computational Linguistics*.

van Cranenburgh, A. and Bod, R. (2013). Discontinuous parsing with an efficient and accurate dop model. *Proc. of International Conference on Parsing Technologies*.

Versley, Y. (2014a). Experiments with easy-first nonprojective constituent parsing. In *Proc. of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*.

Versley, Y. (2014b). Incorporating semi-supervised features into discontinuous easy-first constituent parsing. *CoRR*, abs/1409.3813.

Vijay-Shanker, K., Weir, D. J., and Joshi, A. K. (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. of the Annual Meeting on Association for Computational Linguistics*.

Xia, F. and Palmer, M. (2001). Converting dependency structures to phrase structures. In *Proc. of the First International Conference on Human Language Technology Research*.

Xia, F., Rambow, O., Bhatt, R., Palmer, M., and Misra Sharma, D. (2008). Towards a multi-representational treebank. *LOT Occasional Series*.

Zhu, M., Zhang, Y., Chen, W., Zhang, M., and Zhu, J. (2013). Fast and accurate shift-reduce constituent parsing. In *Proc. of Annual Meeting of the Association for Computational Linguistics*.