Current Trends in Data-Driven Dependency Parsing

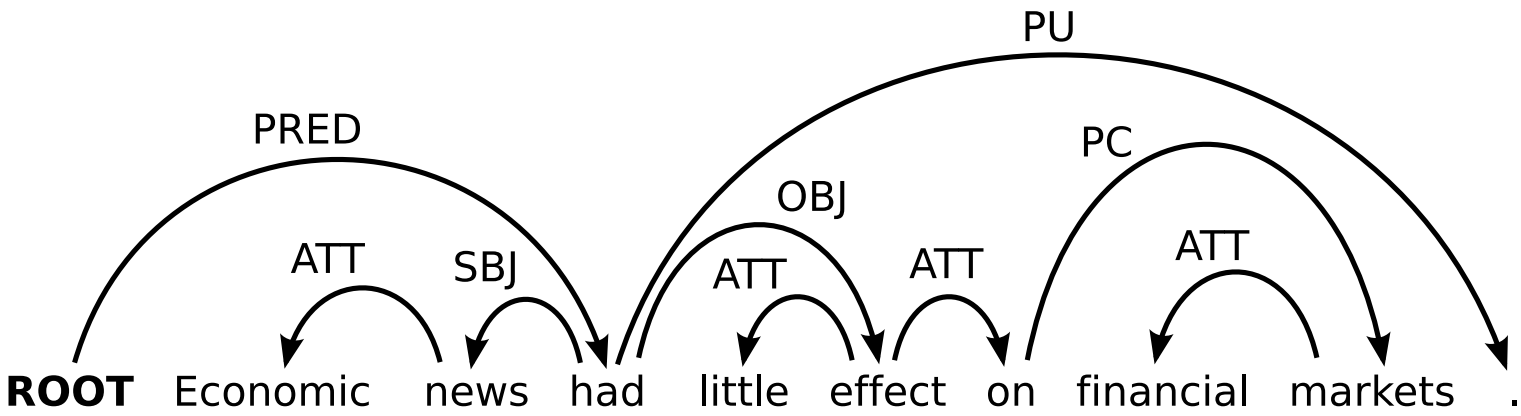
Joakim Nivre

Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Joint work with Ryan McDonald, Google Research

Introduction

- Syntactic parsing of natural language:
  - Who does what to whom?
- Dependency-based syntactic representations
  - give a transparent encoding of predicate-argument structure
  - can be parsed very efficiently
  - can be learned using (simple) data-driven models



Data-Driven Dependency Parsing

- Very active area of research during the last decade:
  - New models proposed and evaluated
  - Improved accuracy and efficiency for a range of languages
  - CoNLL shared tasks 2006 and 2007 (19 languages, 42 systems)
- Time for reflection and analysis?
  - How can we formally categorize different approaches?
  - Can we characterize their errors through an empirical analysis?
  - Can we benefit from this analysis to build improved parsers?

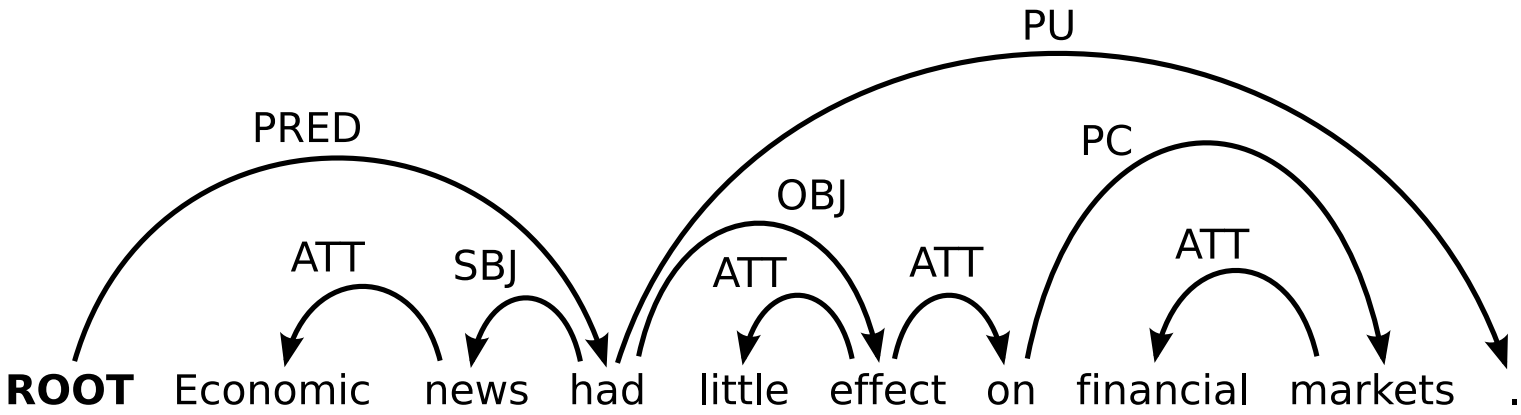
Structure of This Talk

- Data-driven dependency parsing
- Two models of dependency parsing:
  - Graph-based parsing
  - Transition-based parsing
- Analyzing the models:
  - Theoretical analysis
  - Empirical error analysis
- Improving the models
  - System integration
  - Novel approaches
- Conclusion

Data-Driven Dependency Parsing

Dependency Syntax

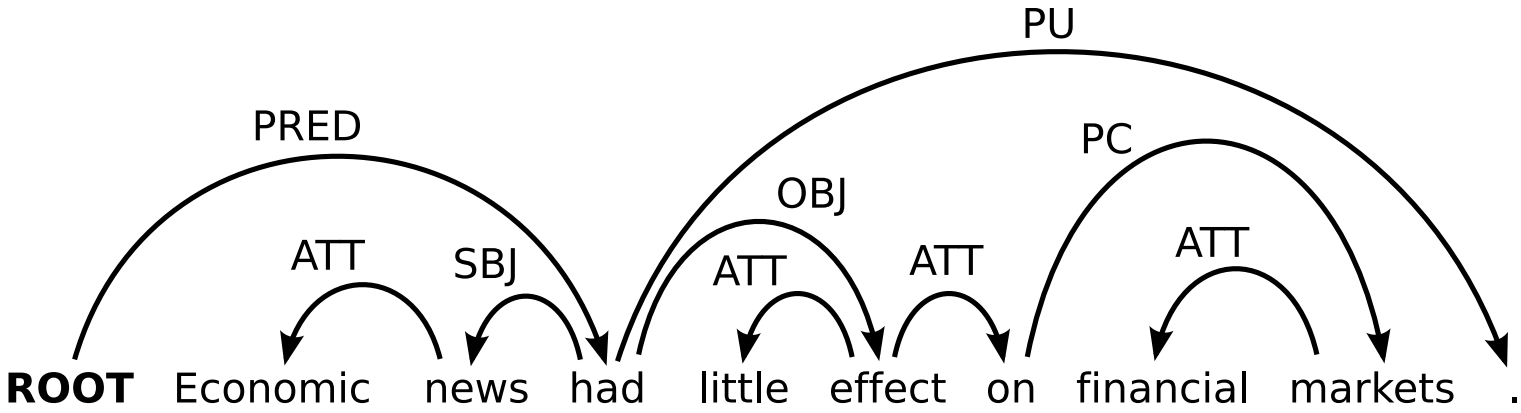
- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies
- Many different theoretical frameworks



Dependency Trees

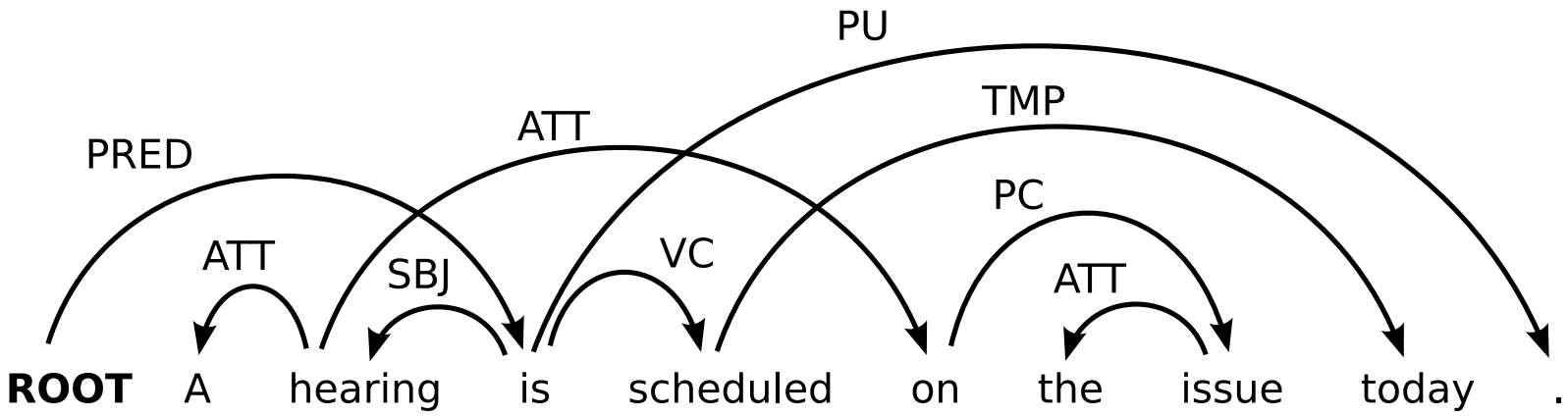
- A dependency structure for a sentence $S = w_1, \ldots, w_n$ is a labeled directed tree $T = (V, A)$, consisting of
  - a set V of nodes, labeled with words w_i (including ROOT)
  - a set A of arcs (w_i, w_j, l), labeled with dependency types $l \in L$

  with the node labeled ROOT as the unique root



Projectivity

- A dependency tree T is projective iff every word has a contiguous projection in S
  - Projection of w = the set of words dominated by w
- Most theoretical frameworks do not assume projectivity
- Non-projective structures needed to account for
  - long-distance dependencies
  - free word order



Data-Driven Dependency Parsing

- Dependency parsing:
  - Input: Sentence $S = w_1, \ldots, w_n$
  - Output: Dependency tree T
- Data-driven parsing:
  - Parsing model learned from data (treebank)
  - May or may not involve a formal grammar
- Two basic problems:
  - Learning: Estimate model parameters from training data
  - Inference: Find best analysis according to model

Two Models of Dependency Parsing

Two Models of Dependency Parsing

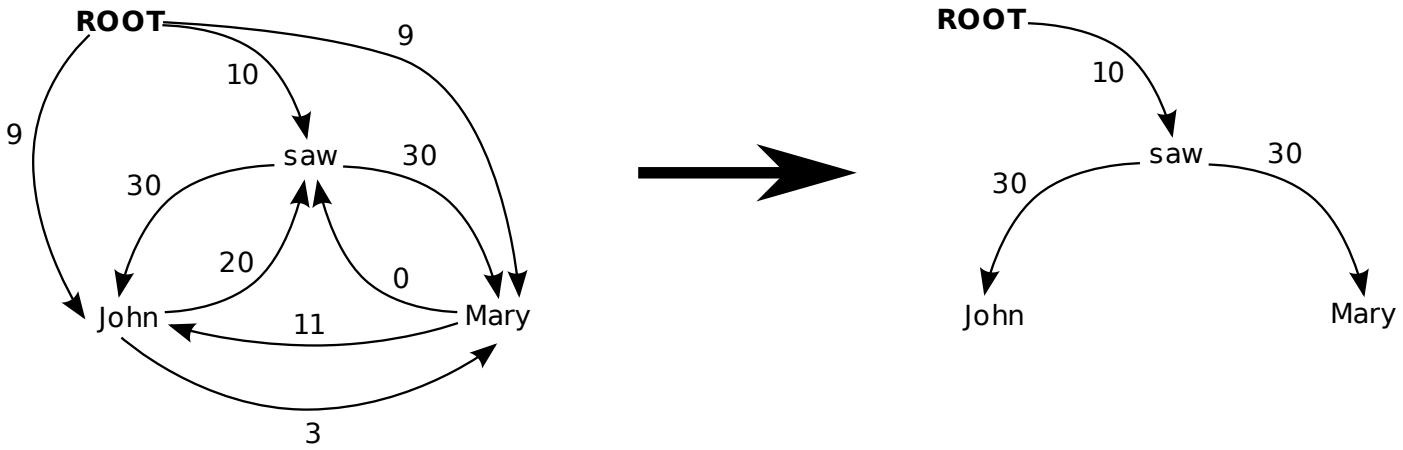
- Graph-based [Eisner 1996, McDonald et al. 2005a]:
  - Parameterize parsing model by dependency arcs
  - Learn to predict entire trees given the input
  - Predict new trees using spanning tree algorithms
- Transition-based [Yamada and Matsumoto 2003, Nivre et al. 2004]:
  - Parameterize parsing model by state transitions
  - Learn to predict transitions given the input and a history
  - Predict new trees using greedy inference
- CoNLL shared task 2006:
  - 6 graph-based systems; best score: 80.83 [McDonald et al. 2006]
  - 10 transition-based systems; best score: 80.75 [Nivre et al. 2006]

Graph-Based Parsing

- For input sentence S define a graph $G_S = (V_S, A_S)$, where
  - $V_S = \{w_0, w_1, \ldots, w_n\}$
  - $A_S = \{(w_i, w_j, l) \mid w_i, w_j \in V \text{ and } l \in L\}$
- Key observation:
  - Valid dependency trees for S = directed spanning trees of G_S
- Score of dependency tree T factors by subgraphs $G_1, \ldots, G_m$:
  - $s(T) = \sum_{i=1}^{m} s(G_i)$
- Learning:
  - Scoring function $s(G_i)$ for subgraphs $G_i \in T$
- Inference:
  - Search for maximum spanning tree T^* of G_S given $s(G_i)$

Parameterizing Graph-Based Parsing

- First-order (arc-factored) model:
  - $s(T = (V, A)) = \sum_{(w_i, w_j, l) \in A} s(w_i, w_j, l)$
  - Exact inference in $O(n^2)$ time for non-projective trees using the Chu-Liu-Edmonds algorithm [McDonald et al. 2005b]



Parameterizing Graph-Based Parsing

- Higher-order models [McDonald and Pereira 2006, Carreras 2007]:
  - Subgraphs G_i involving a (small) number of arcs
  - Intractable in non-projective case [McDonald and Satta 2007]
  - Exact inference in $O(n^3)$ time for projective trees with second-order model using Eisner's algorithm [Eisner 1996]
- Efficient parsing requires that scores factor by small subgraphs

Learning Graph-Based Models

- Typical scoring function:
  - $s(G_i) = \mathbf{w} \cdot \mathbf{f}(G_i)$

  where
  - $\mathbf{f}(G_i)$ = high-dimensional feature vector over subgraphs
  - $\mathbf{w}$ = weight vector [$\mathbf{w}_j$ = weight of feature $\mathbf{f}_j(G_i)$]
- Structured learning [McDonald et al. 2005a, Smith and Johnson 2007]:
  - Learn weights that maximize the score of the correct dependency tree for every sentence in the training set
- Learning is global (trees), but features are local (subgraphs)

Transition-Based Parsing

- A transition system for dependency parsing defines
  - a set C of parser configurations
  - a set D of transitions, each a function $d : C \rightarrow C$
  - initial configuration and terminal configurations for sentence S
- Key idea:
  - Valid dependency trees for S defined by terminating transition sequences $D_{1,m} = d_1(c_1), \ldots, d_m(c_m)$
- Score of $D_{1,m}$ factors by configuration-transition pairs (c_i, d_i):
  - $s(D_{1,m}) = \sum_{i=1}^{m} s(c_i, d_i)$
- Learning:
  - Scoring function $s(c_i, d_i)$ for $d_i(c_i) \in D_{1,m}$
- Inference:
  - Search for highest scoring sequence $D^*_{1,m}$ given $s(c_i, d_i)$

Example: Arc-Eager Projective Parsing

Configuration: (S, B, A) $[S = \text{Stack}, B = \text{Buffer}, A = \text{Arcs}]$

Initial: $([\text{ROOT}], [w_1, \ldots, w_n], \{\ \})$

Terminal: $(S, [\ ], A)$

Shift:

$$([\ldots, w_i], [w_j, \ldots], A) \implies ([\ldots, w_i, w_j], [\ldots], A)$$

Reduce:

$$([\ldots, w_i], B, A) \implies ([\ldots], B, A)$$

Right-Arc(l):

$$([\ldots, w_i], [w_j, \ldots], A) \implies ([\ldots, w_i, w_j], [\ldots], A \cup \{(w_i, w_j, l)\})$$

Left-Arc(l):

$$([\ldots, w_i], [w_j, \ldots], A) \implies ([\ldots], [w_j, \ldots], A \cup \{(w_j, w_i, l)\})$$

Inference for Transition-Based Parsing

- Exact inference intractable for standard transition systems
- Common inference strategies:
  - Deterministic [Yamada and Matsumoto 2003, Nivre et al. 2004]
  - Beam search [Johansson and Nugues 2006, Titov and Henderson 2007]
  - Complexity given by upper bound on transition sequence length
- Transition systems:
  - Projective $O(n)$ [Yamada and Matsumoto 2003, Nivre 2003]
  - Limited non-projective $O(n)$ [Attardi 2006, Nivre 2007]
  - Unrestricted non-projective $O(n^2)$ [Nivre 2008, Nivre 2009]
- Efficient parsing requires approximate inference

Learning for Transition-Based Parsing

- Typical scoring function:
  - $s(c_i, d_i) = \mathbf{w} \cdot \mathbf{f}(c_i, d_i)$

  where
  - $\mathbf{f}(c_i, d_i)$ = feature vector over configuration c_i and transition d_i
  - $\mathbf{w}$ = weight vector [$\mathbf{w}_j$ = weight of feature $\mathbf{f}_j(c_i, d_i)$]
- Simple classification problem:
  - Learn weights that maximize the score of the correct transition out of every configuration in the training set
  - Configurations represent derivation history, including partially built dependency tree
- Learning is local, but features are global

Summing Up

- Graph-based models:
  - Global learning
  - Exact (or nearly exact) inference
  - Local graph-based features
- Transition-based models:
  - Local learning
  - Greedy (often deterministic) inference
  - Global history-based features

Analyzing the Models

Analyzing the Models

- Motivation:
  - Two radically different models with very similar performance
  - Are the errors performed by both models the same or different?
  - If different, can they be explained by properties of the models?
  - If explained, can they be eliminated by improving the models?
- Case study of the best systems in the CoNLL-X shared task:
  - MSTParser [McDonald et al. 2006] – graph-based
  - MaltParser [Nivre et al. 2006] – transition-based

Comparison

- Inference:
  - Nearly exhaustive (MST)
  - Greedy deterministic (Malt)
- Learning:
  - Global structure learning (MST)
  - Local decision learning (Malt)
- Feature scope:
  - Local graph features (MST)
  - Rich decision history (Malt)
- Fundamental trade-off:
  - Rich feature space – global learning and inference

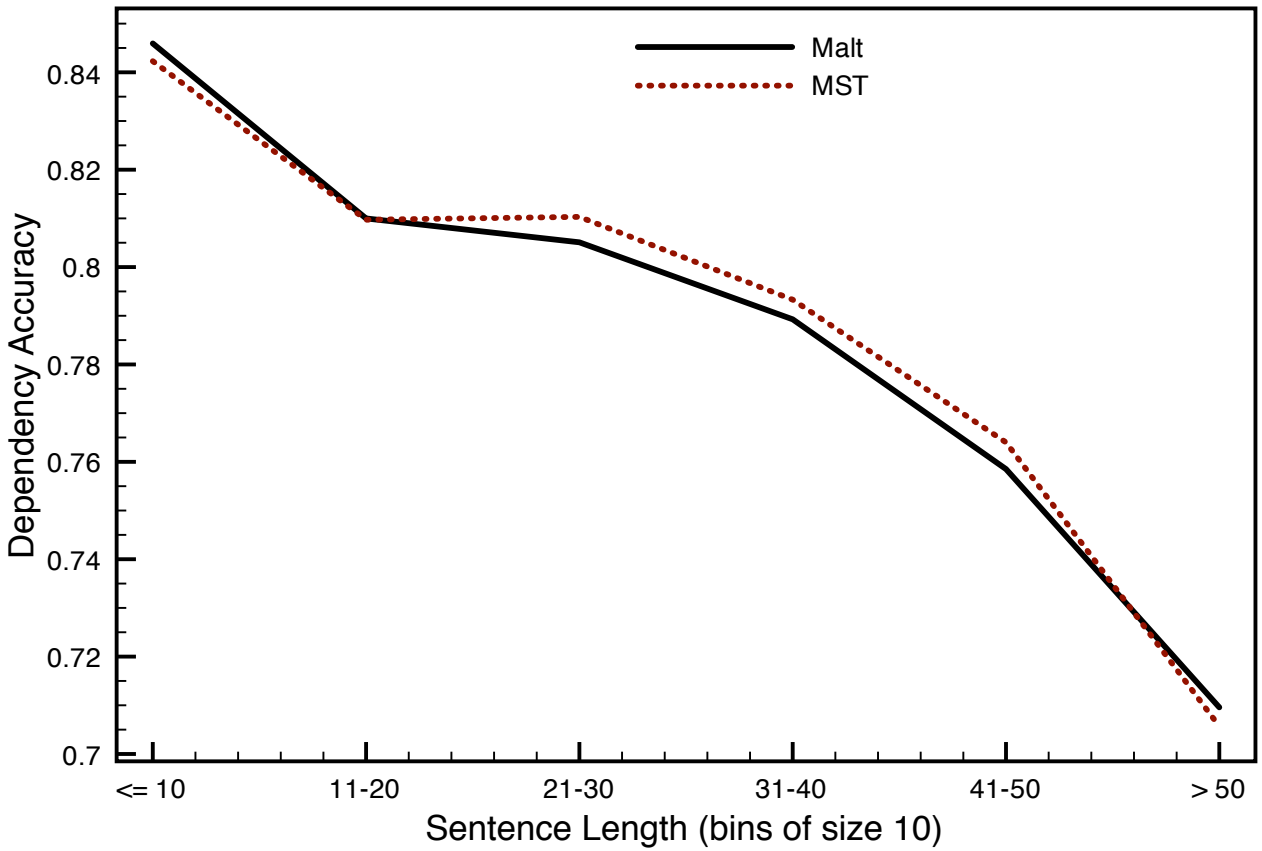
Overall Parsing Accuracy

Language	MST	Malt
Arabic	**66.91**	66.71
Bulgarian	**87.57**	87.41
Chinese	85.90	**86.92**
Czech	**80.18**	78.42
Danish	**84.79**	84.77
Dutch	**79.19**	78.59
German	**87.34**	85.82
Japanese	90.71	**91.65**
Portuguese	86.82	**87.60**
Slovene	**73.44**	70.30
Spanish	**82.25**	81.29
Swedish	82.55	**84.58**
Turkish	63.19	**65.68**
Average	**80.83**	80.75

Error Analysis

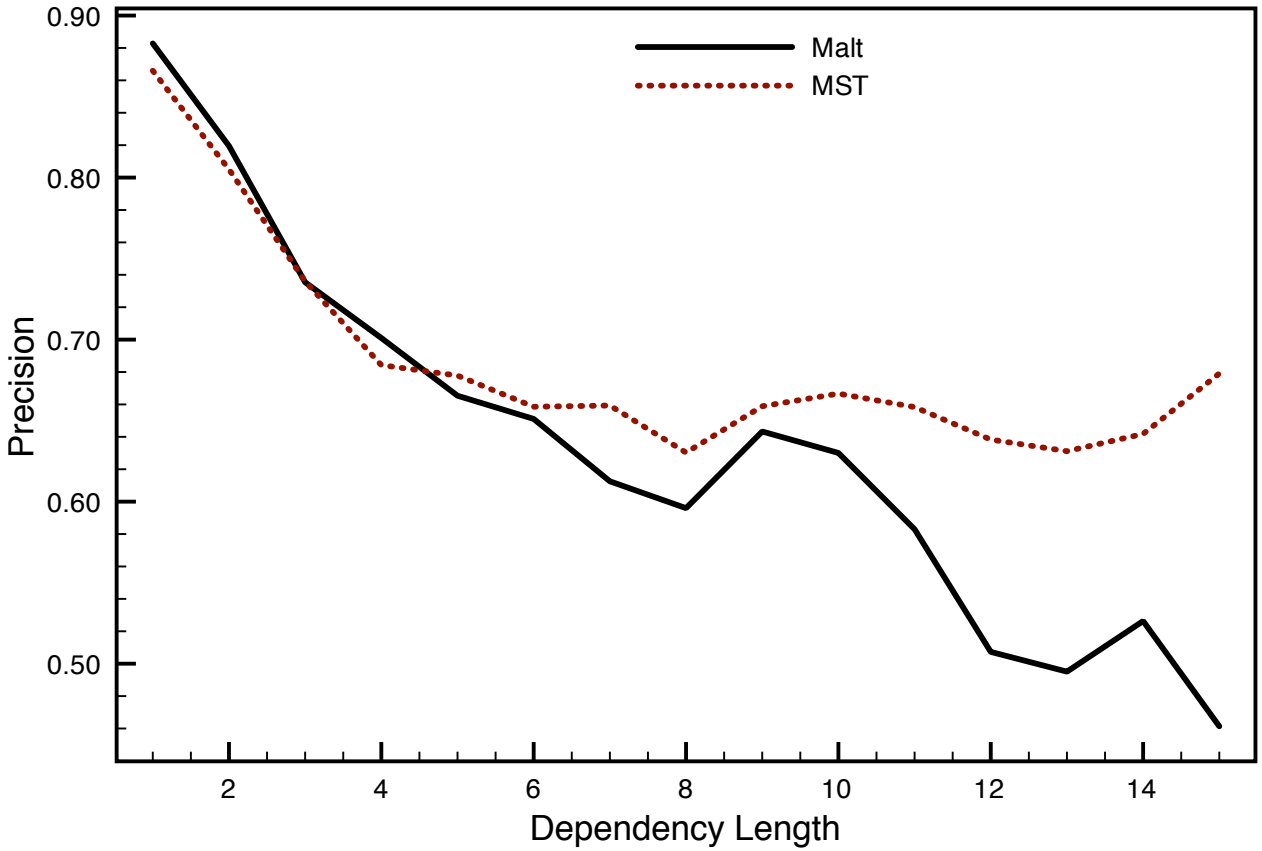
- Aim:
  - Relate parsing errors to linguistic and structural properties of sentences and their dependency trees
- Statistics:
  - Labeled accuracy, precision and recall
  - Based on the test sets from the CoNLL-X shared task
  - Statistics averaged over all 13 languages

Sentence Length – Accuracy



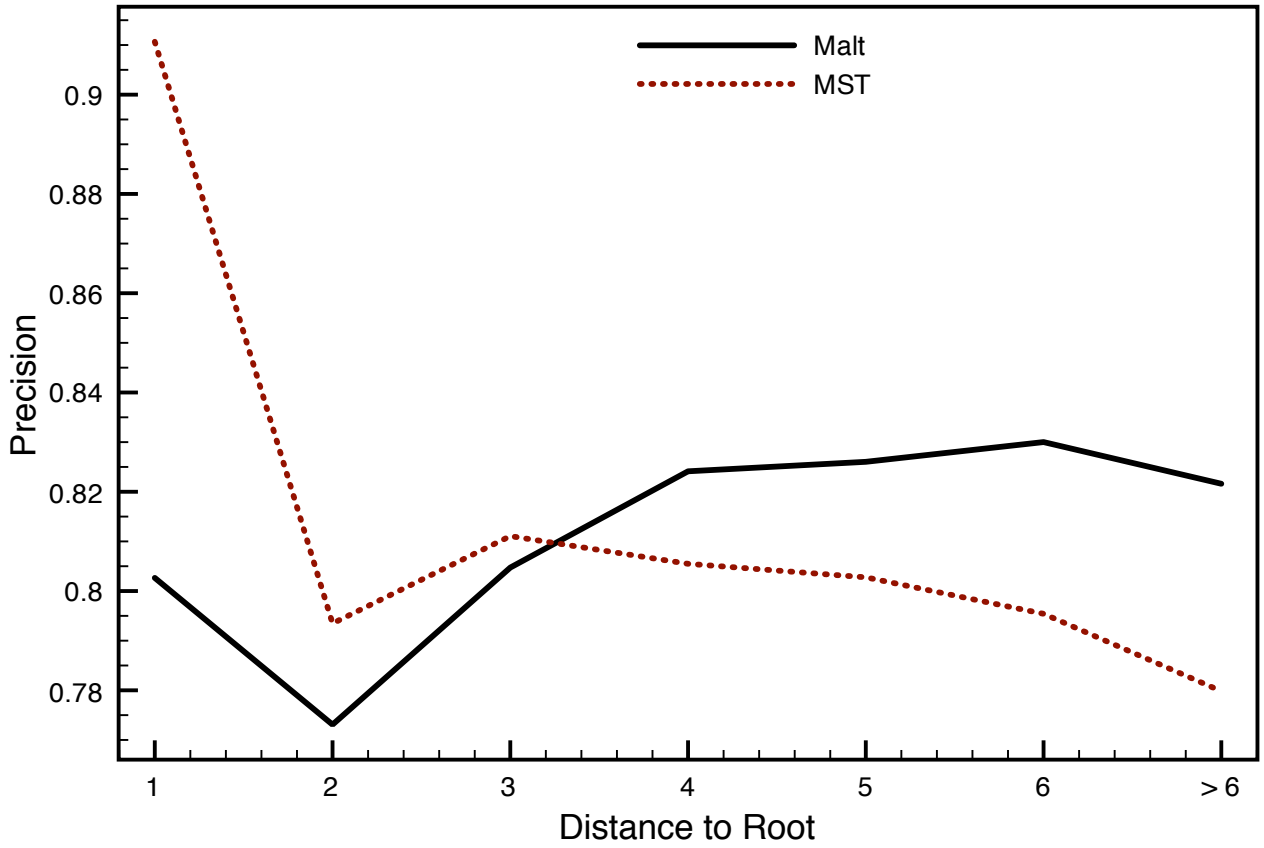
- MaltParser more accurate for short sentences (1–10 words)
- MSTParser degrades less with increasing sentence length

Dependency Length – Precision



- MaltParser more precise for short dependencies (1–3 words)
- MSTParser degrades less with increasing dependency length

Tree Depth (Distance to Root) – Precision



- MSTParser more precise for root dependents, then constant
- MaltParser improves with increasing depth (up to 7 arcs)

Part of Speech – Accuracy

Part of Speech	MST	Malt
Verb	**82.6**	81.9
Noun	80.0	**80.7**
Pronoun	88.4	**89.2**
Adjective	**89.1**	87.9
Adverb	**78.3**	77.4
Adposition	**69.9**	68.8
Conjunction	**73.1**	69.8

- MSTParser more accurate for verbs, adjectives, adverbs, adpositions, and conjunctions
- MaltParser more accurate for nouns and pronouns

Discussion

- Many of the results are indicative of the fundamental trade-off: global learning/inference versus rich features
- Global learning/inference improves decisions for long sentences and dependencies and those near the top of trees
- Rich features improve decisions for short sentences and dependencies and those near the leaves of trees
- Important question:
  - How do we use this to improve parser performance?
- Oracle experiments:
  - Tree-based selection: 81% $\rightarrow$ 85%
  - Arc-based selection: 81% $\rightarrow$ 87%

Improving the Models

System Integration – Voting

- Ensemble systems:
  - Given the outputs of *m* systems, choose the majority output
  - Black box combination – requires at least three systems
- Greedy head word selection [Zeman and Žabokrtský 2005]:
  - Ensemble of (up to) seven parsers
  - 2% improvement over best single parser on Czech
  - Problem: Output may not be a tree
- Parser combination by reparsing [Sagae and Lavie 2006]:
  - Use weighted votes as arc scores in a graph-based model
  - Use maximum spanning tree algorithms for inference
  - Ensemble of graph-based and transition-based parsers
  - 1.7% improvement over best single parser on English

System Integration – Stacking

- Feature-based integration by stacking:
  - Use output of one system as input features for another
  - Allows one system to learn relative to the behavior of another
  - More efficient than reparsing – only requires two parsers
- Guided parsing [Nivre and McDonald 2008, Torres Martins et al. 2008]:
  - Train guided parser A_B (A guided by B) on

$$D^B = \{(S_1, T_1, T_1^B), \ldots, (S_m, T_m, T_m^B)\}$$

where

- T_i = gold standard tree for sentence S_i,
- T_i^B = output of parser B on S_i (using cross-validation)

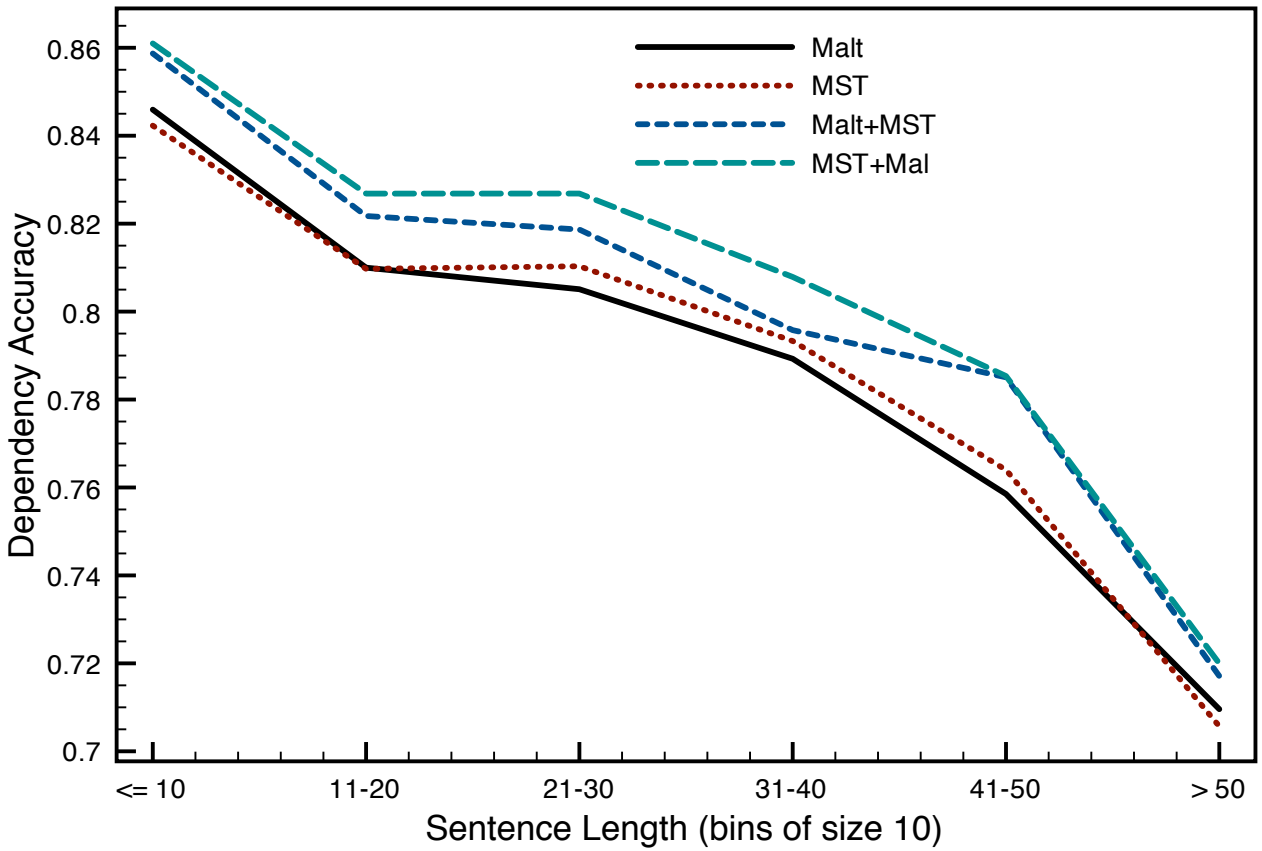
Guided Parsers: MST_{Malt} and Malt_{MST}

- New features for MST_{Malt} [defined on arc (w_i, w_j, l); $* = \text{any}$]:
  - Is $(w_i, w_j, *)$ in T_i^{Malt}?
  - Is (w_i, w_j, l) in T_i^{Malt}?
  - Is $(w_i, w_j, *)$ not in T_i^{Malt}?
  - Is (w_i, w_j, l) not in T_i^{Malt}?
  - Identity of l' such that $(*, w_j, l')$ is in T_i^{Malt}?
  - Identity of l' such that (w_i, w_j, l') is in T_i^{Malt}?
- New features for Malt_{MST} [defined on configuration c; $* = \text{any}$]:
  - Is $(\sigma_c^0, \beta_c^0, *)$ in T_i^{MST}?
  - Is $(\beta_c^0, \sigma_c^0, *)$ in T_i^{MST}?
  - Head direction for σ_c^0 in T_i^{MST} (left/right/root)
  - Head direction for β_c^0 in T_i^{MST} (left/right/root)
  - Identity of l such that $(*, \sigma_c^0, l)$ is in T_i^{MST}?
  - Identity of l such that $(*, \beta_c^0, l)$ is in T_i^{MST}?

Experimental Results

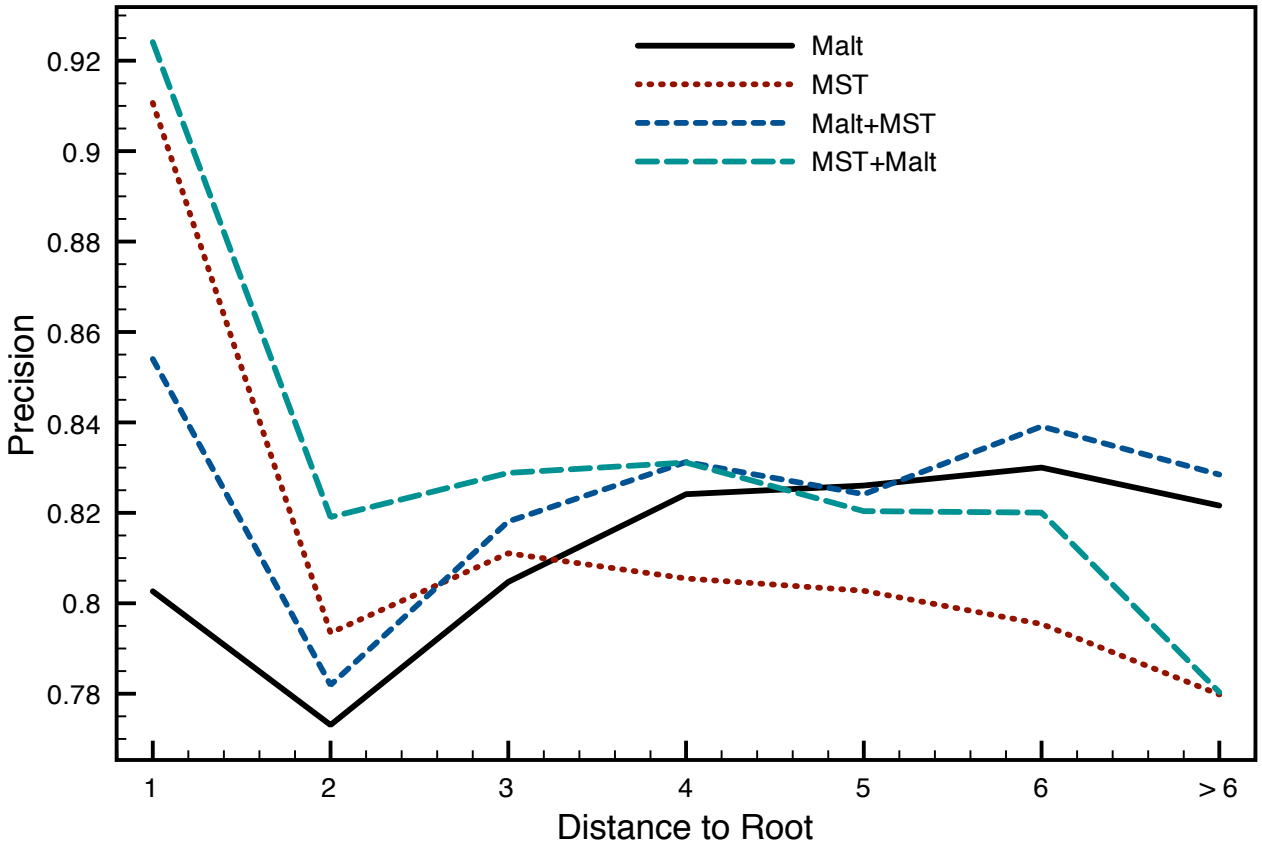
Language	MST	MST_{Malt}	Malt	Malt_{MST}
Arabic	66.91	**68.64** (+1.73)	66.71	67.80 (+1.09)
Bulgarian	87.57	**89.05** (+1.48)	87.41	88.59 (+1.18)
Chinese	85.90	**88.43** (+2.53)	86.92	87.44 (+0.52)
Czech	80.18	**82.26** (+2.08)	78.42	81.18 (+2.76)
Danish	84.79	**86.67** (+1.88)	84.77	85.43 (+0.66)
Dutch	79.19	**81.63** (+2.44)	78.59	79.91 (+1.32)
German	87.34	**88.46** (+1.12)	85.82	87.66 (+1.84)
Japanese	90.71	91.43 (+0.72)	91.65	**92.20** (+0.55)
Portuguese	86.82	87.50 (+0.68)	87.60	**88.64** (+1.04)
Slovene	73.44	**75.94** (+2.50)	70.30	74.24 (+3.94)
Spanish	82.25	**83.99** (+1.74)	81.29	82.41 (+1.12)
Swedish	82.55	**84.66** (+2.11)	84.58	84.31 (–0.27)
Turkish	63.19	64.29 (+1.10)	65.58	**66.28** (+0.70)
Average	80.83	**82.53** (+1.70)	80.75	82.01 (+1.27)

Sentence Length – Accuracy



- MST_{Malt} improves most for shorter sentences
- Malt_{MST} improves most for longer sentences

Tree Depth (Distance to Root) – Precision



- MST_{Malt} improves most for deeper dependencies
- Malt_{MST} improves most for dependencies near the root

Part of Speech – Accuracy

Part of Speech	MST	MST_{Malt}	Malt	Malt_{MST}
Verb	82.6	**85.1** (2.5)	81.9	84.3 (2.4)
Noun	80.0	81.7 (1.7)	80.7	**81.9** (1.2)
Pronoun	88.4	**89.4** (1.0)	89.2	89.3 (0.1)
Adjective	89.1	**89.6** (0.5)	87.9	89.0 (1.1)
Adverb	78.3	**79.6** (1.3)	77.4	78.1 (0.7)
Adposition	69.9	**71.5** (1.6)	68.8	70.7 (1.9)
Conjunction	73.1	**74.9** (1.8)	69.8	72.5 (2.7)

- Tendencies:
  - Parser with worst performance improves the most
  - Guided parser outperforms both base parser and guide parser

Conclusion – Stacking

- Benefits of feature-based integration:
  - Guided parser can learn when to trust guide parser
  - Improvement in weak areas due to guide features
  - Improvement in strong areas due to improvement in weak areas
- Limitations:
  - Error propagation eliminates potential improvement for Malt
  - Feature scope remains rather limited for MST

Exploring the Fundamental Trade-Off

- Graph-based parsing with global features:
  - Reranking [Hall 2007]
  - Gibbs sampling [Nakagawa 2007]
  - Loopy belief propagation [Smith and Eisner 2008]
  - Integer linear programming [Riedel et al. 2006, Martins et al. 2009]
- Transition-based parsing with global learning and inference:
  - Beam search [Johansson and Nugues 2006, Titov and Henderson 2007]
  - Global structure learning [Zhang and Clark 2008]

Conclusion

Conclusion

- Two models of dependency parsing:
  - Graph-based – global learning/inference, local features
  - Transition-based – local learning/inference, global features
- Model properties are reflected in error distributions:
  - Graph-based – less accurate on local structures
  - Transition-based – less accurate on global structures
- Analysis informs (and explains) improvements:
  - System integration – exploit complementary strengths
  - New models – better trade-off between global and local aspects

Finally . . .

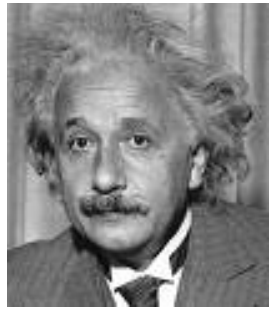
Finally . . .

- Natural language processing – the art of approximation?

Finally . . .

- Natural language processing – the art of approximation?

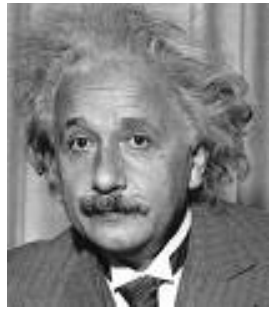
> "Everything should be made as simple as possible, but no simpler." (Albert Einstein)



Finally . . .

- Natural language processing – the art of approximation?

  "Everything should be made as simple as possible, but no simpler." (Albert Einstein)

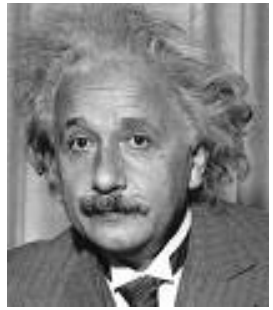


- Empirical error analysis supports theory development

Finally . . .

- Natural language processing – the art of approximation?

  "Everything should be made as simple as possible, but no simpler." (Albert Einstein)



- Empirical error analysis supports theory development

  "The temptation to form premature theories upon insufficient data is the bane of our profession." (Sherlock Holmes)

- Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 166–170.
- Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 957–961.
- Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.
- Keith Hall. 2007. K-best spanning tree parsing. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 392–399.
- Richard Johansson and Pierre Nugues. 2006. Investigating multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 206–210.
- Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 342–350.
- Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 81–88.
- Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pages 122–131.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.

- Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 216–220.
- Tetsuji Nakagawa. 2007. Multilingual dependency parsing using global features. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 952–956.
- Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 950–958.
- Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning*, pages 49–56.
- Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 221–225.
- Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.
- Joakim Nivre. 2007. Incremental non-projective dependency parsing. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 396–403.
- Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.
- Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 351–359.
- Sebastian Riedel, Ruket Çakıcı, and Ivan Meza-Ruiz. 2006. Multi-lingual dependency parsing with incremental integer linear programming. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 226–230.

- Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132.
- David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 145–156.
- Noah A. Smith and Mark Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33:477–491.
- Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pages 144–155.
- André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 157–166.
- Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.
- Daniel Zeman and Zdeněk Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pages 171–178.
- Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 562–571.