Beyond MaltParser

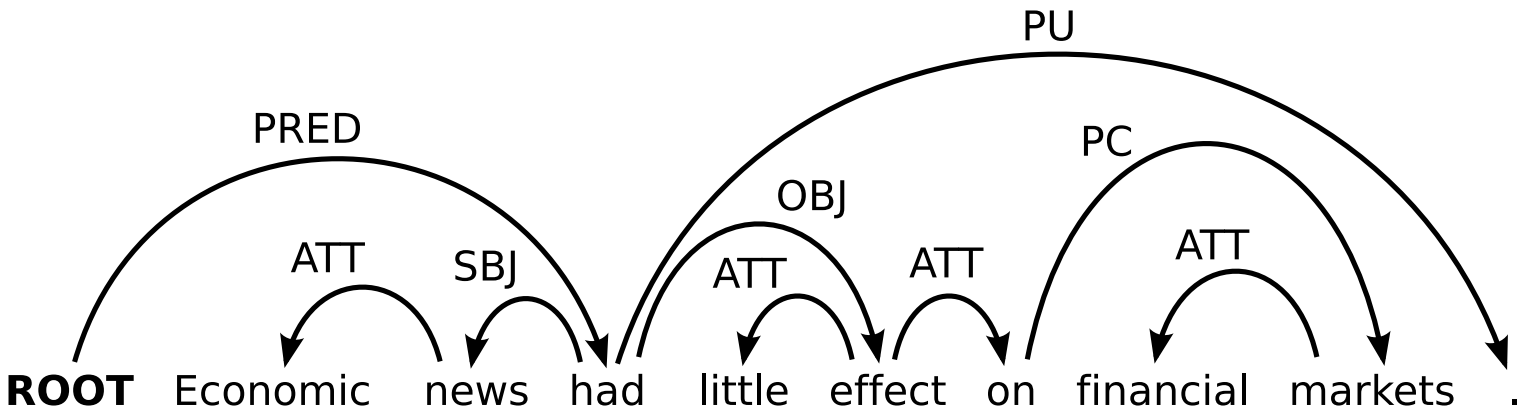
Advances in Transition-Based Dependency Parsing

Joakim Nivre

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

Introduction

- Syntactic parsing of natural language
  - Who does what to whom?
- Dependency-based syntactic representations
  - Long tradition in descriptive and theoretical linguistics
  - Increasingly popular in computational linguistics



Transition-Based Dependency Parsing

- The basic idea:
  - Define a transition system for dependency parsing
  - Learn a model for scoring possible transitions
  - Parse by searching for the optimal transition sequence
- Advantages:
  - Highly efficient parsing with low complexity
  - Rich history-based feature models for disambiguation

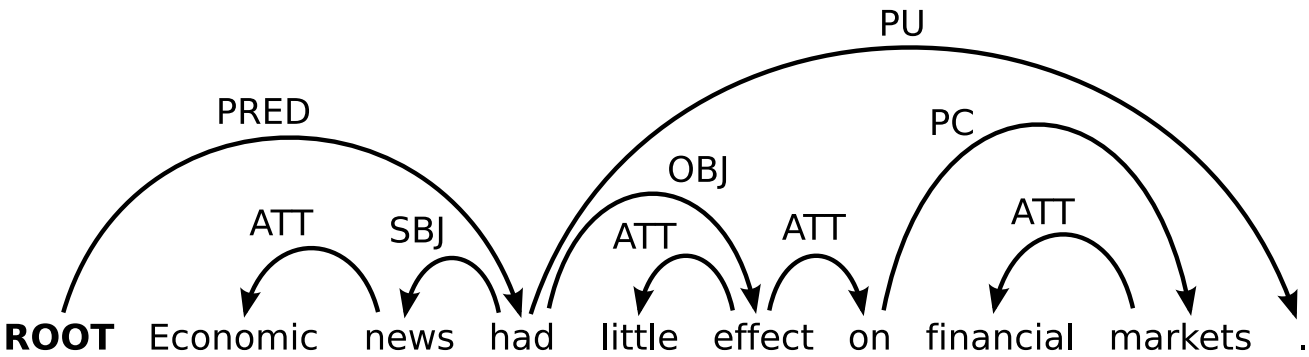
Plan for the Talk

- Basic transition-based dependency parsing
  - Transition system for projective dependency trees
  - Deterministic classifier-based parsing
  - MaltParser “out of the box”
- New developments:
  - Beam search and structured prediction
  - Online reordering for non-projective structures
  - Joint part-of-speech tagging and parsing

Basic Transition-Based Parsing

Dependency Trees

- A dependency tree is a labeled directed tree T with
  - a set V of nodes, labeled with words (including **ROOT**)
  - a set A of arcs, labeled with dependency types
  - a linear precedence order $<$ on V
- Notation:
  - Arc (w_i, l, w_j) connects head w_i to dependent w_j with label l
  - Node w_0 (labeled **ROOT**) is the unique root of the tree



Transition System: Configurations

- A parser configuration is a triple $c = (S, Q, A)$, where
  - S = a stack $[\ldots, w_i]_S$ of partially processed nodes,
  - Q = a queue $[w_j, \ldots]_Q$ of remaining input nodes,
  - A = a set of labeled arcs (w_i, l, w_j).
- Initialization:

  $([w_0]_S, [w_1, \ldots, w_n]_Q, \{\})$

  NB: w_0 = **ROOT**
- Termination:

  $([w_0]_S, []_Q, A)$

Transition System: Transitions

- Left-Arc(l) $\dfrac{([\ldots, w_i, w_j]_S, \quad Q, \quad A)}{([\ldots, w_j]_S, \quad Q, \quad A \cup \{(w_j, l, w_i)\})}$ $[i \neq 0]$
- Right-Arc(l) $\dfrac{([\ldots, w_i, w_j]_S, \quad Q, \quad A)}{([\ldots, w_i]_S, \quad Q, \quad A \cup \{(w_i, l, w_j)\})}$
- Shift $\dfrac{([\ldots]_S, \quad [w_i, \ldots]_Q, \quad A)}{([\ldots, w_i]_S, \quad [\ldots]_Q, \quad A)}$

Example Transition Sequence

$[\textbf{ROOT}]_S$ [Economic, news, had, little, effect, on, financial, markets, .]$_Q$

ROOT Economic news had little effect on financial markets .

Example Transition Sequence

[**ROOT**, Economic]$_S$ [news, had, little, effect, on, financial, markets, .]$_Q$

ROOT Economic news had little effect on financial markets .

Example Transition Sequence

[**ROOT**, Economic, news]$_S$ [had, little, effect, on, financial, markets, .]$_Q$

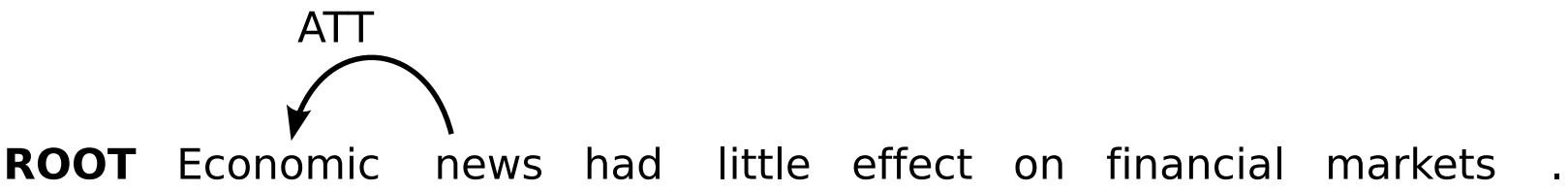
ROOT Economic news had little effect on financial markets .

Example Transition Sequence

[**ROOT**, news]$_S$ [had, little, effect, on, financial, markets, .]$_Q$

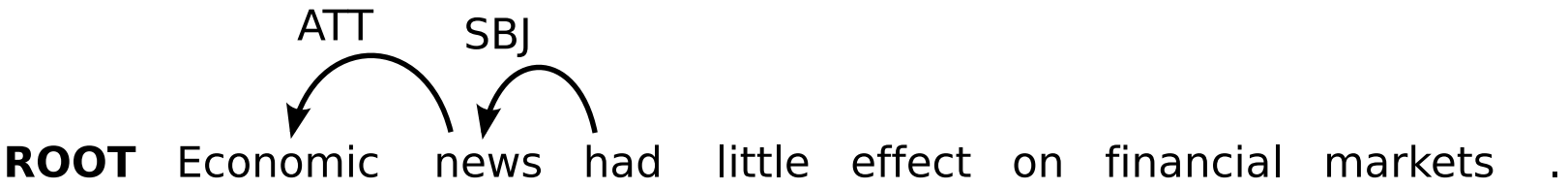
Example Transition Sequence

[**ROOT**, news, had]$_S$ [little, effect, on, financial, markets, .]$_Q$



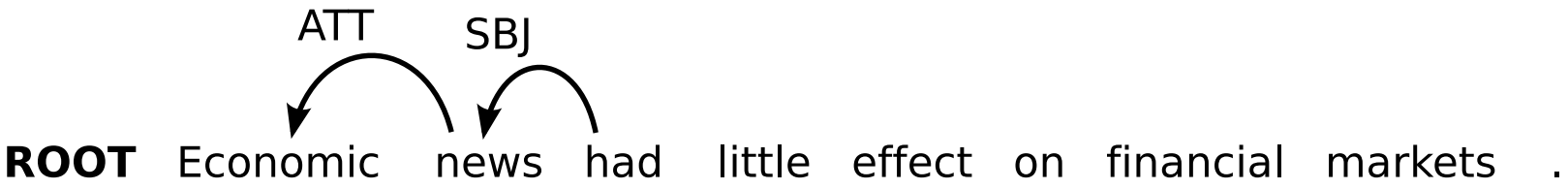
Example Transition Sequence

[**ROOT**, had]$_S$ [little, effect, on, financial, markets, .]$_Q$



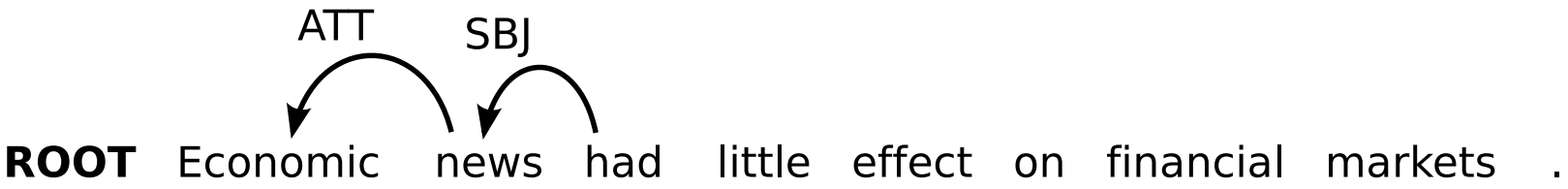
Example Transition Sequence

[**ROOT**, had, little]$_S$ [effect, on, financial, markets, .]$_Q$



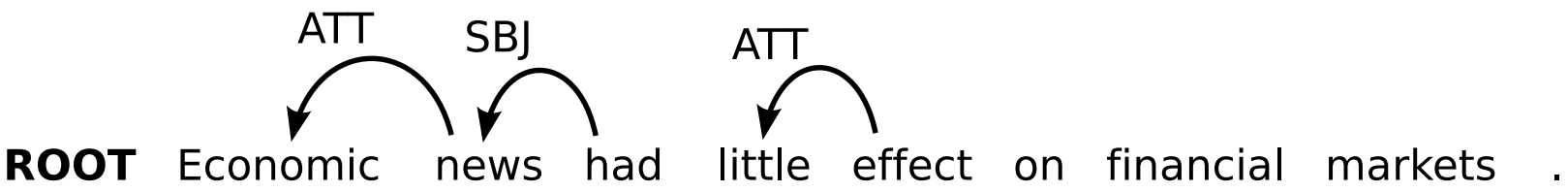
Example Transition Sequence

[**ROOT**, had, little, effect]$_S$ [on, financial, markets, .]$_Q$



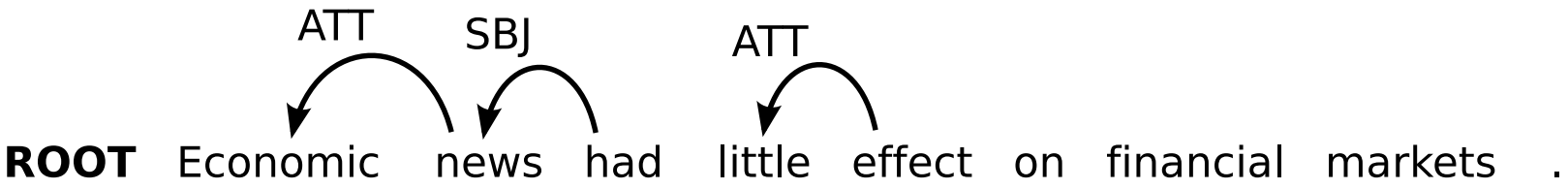
Example Transition Sequence

[**ROOT**, had, effect]$_S$ [on, financial, markets, .]$_Q$



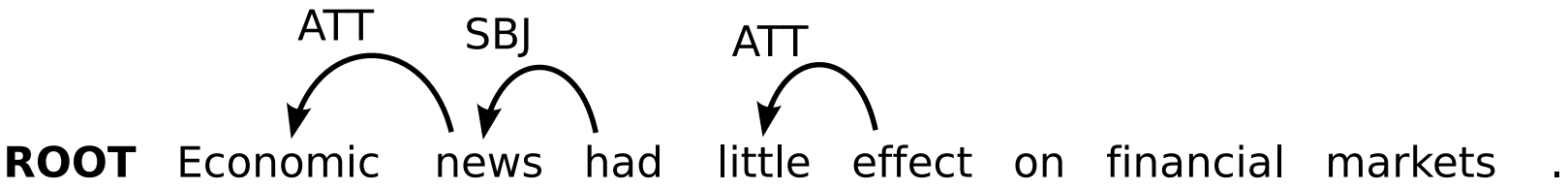
Example Transition Sequence

[**ROOT**, had, effect, on]$_S$ [financial, markets, .]$_Q$



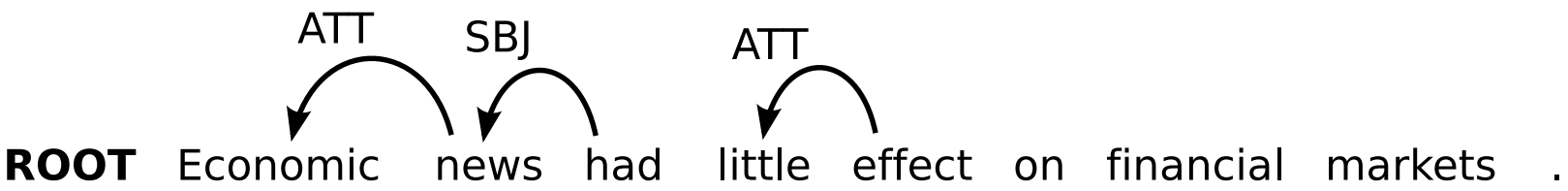
Example Transition Sequence

[**ROOT**, had, effect, on, financial]$_S$ [markets, .]$_Q$



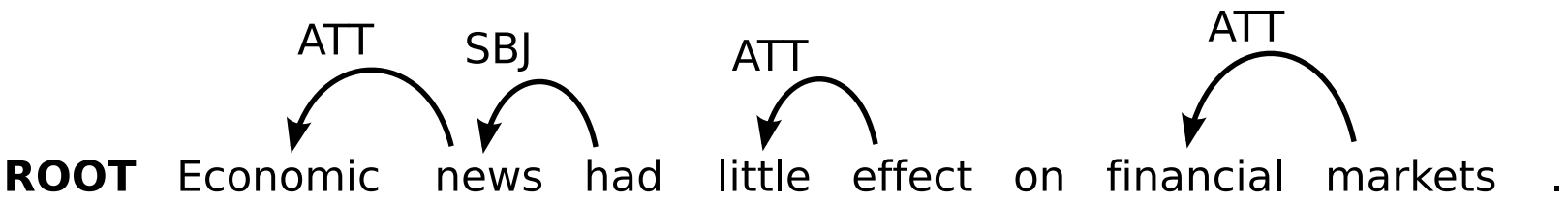
Example Transition Sequence

[**ROOT**, had, effect, on, financial, markets]$_S$ [.]$_Q$



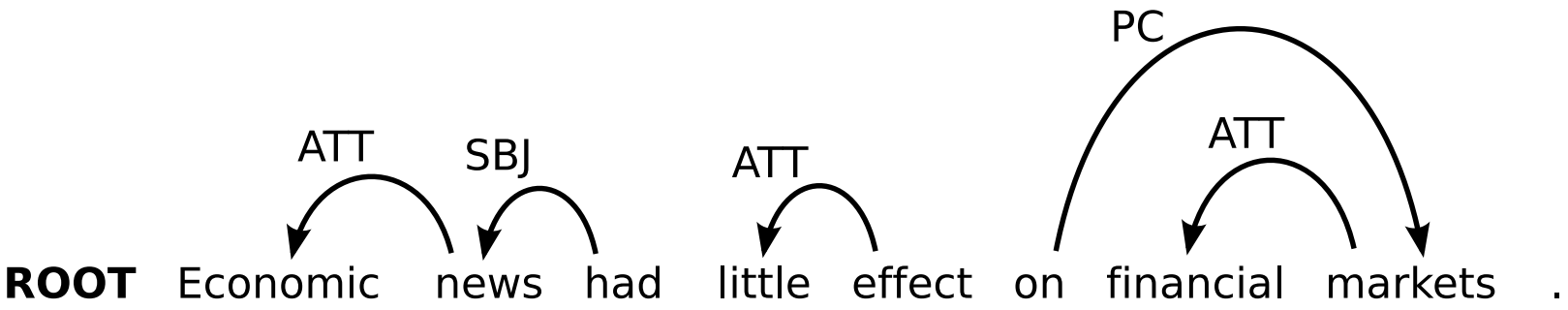
Example Transition Sequence

[**ROOT**, had, effect, on, markets]$_S$ [.]$_Q$



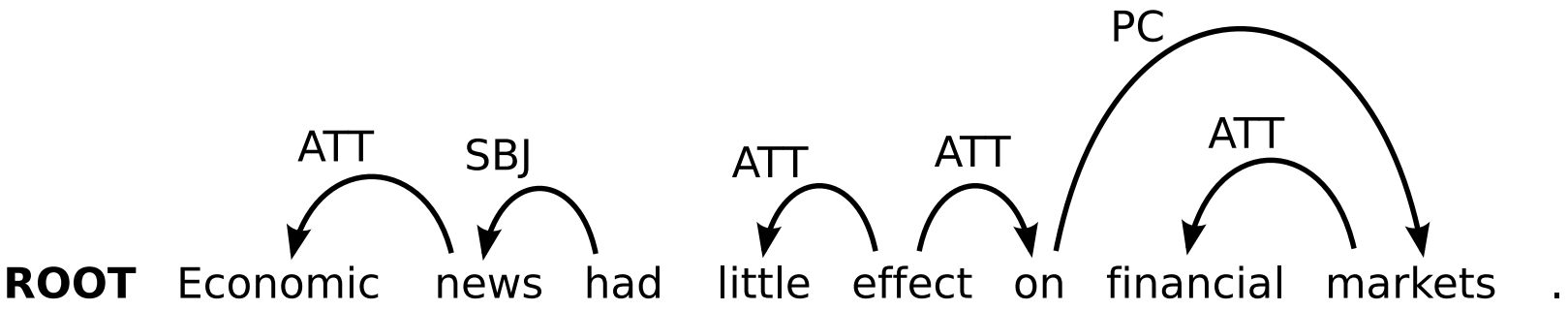
Example Transition Sequence

[**ROOT**, had, effect, on]$_S$ [.]$_Q$



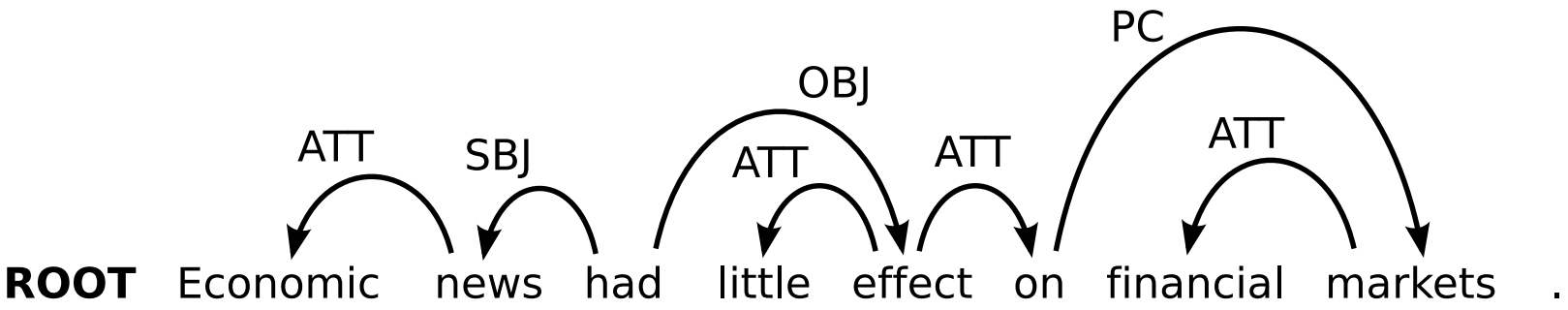
Example Transition Sequence

[**ROOT**, had, effect]$_S$ [.]$_Q$



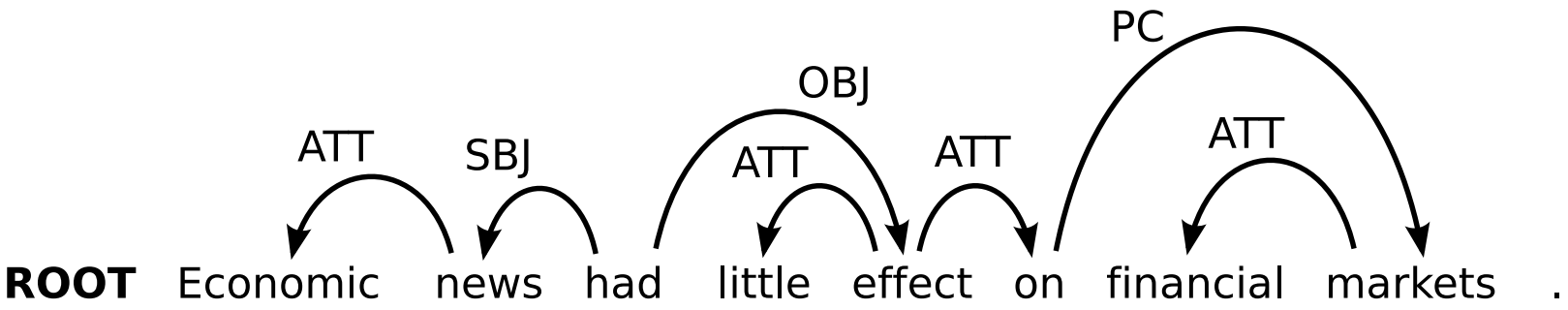
Example Transition Sequence

$[\textbf{ROOT}, \text{had}]_S \; [.]_Q$



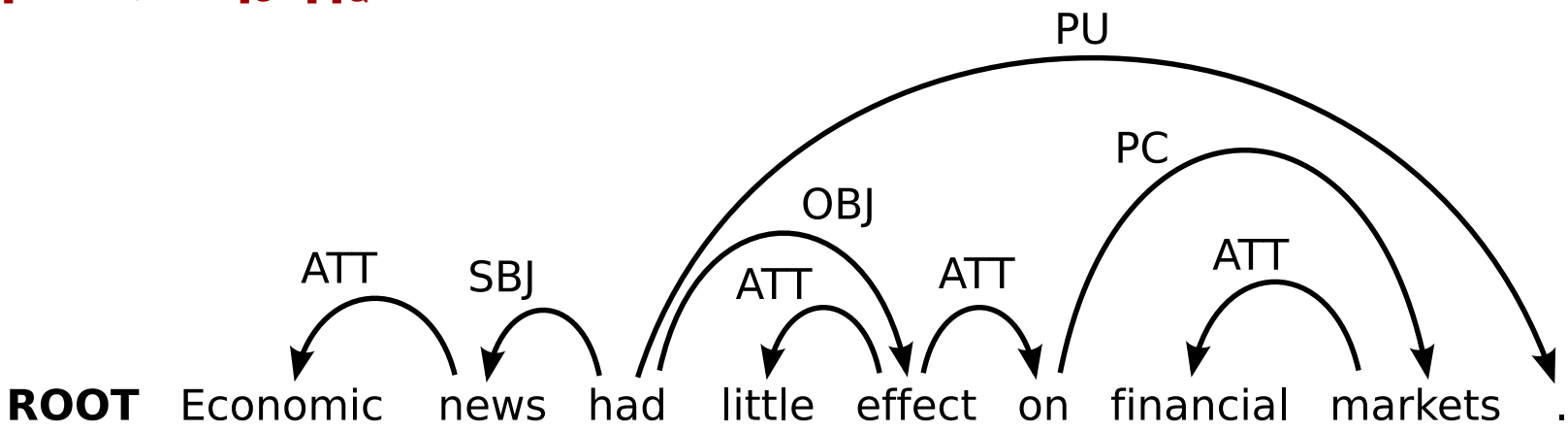
Example Transition Sequence

[**ROOT**, had, .]$_S$ [ ]$_Q$



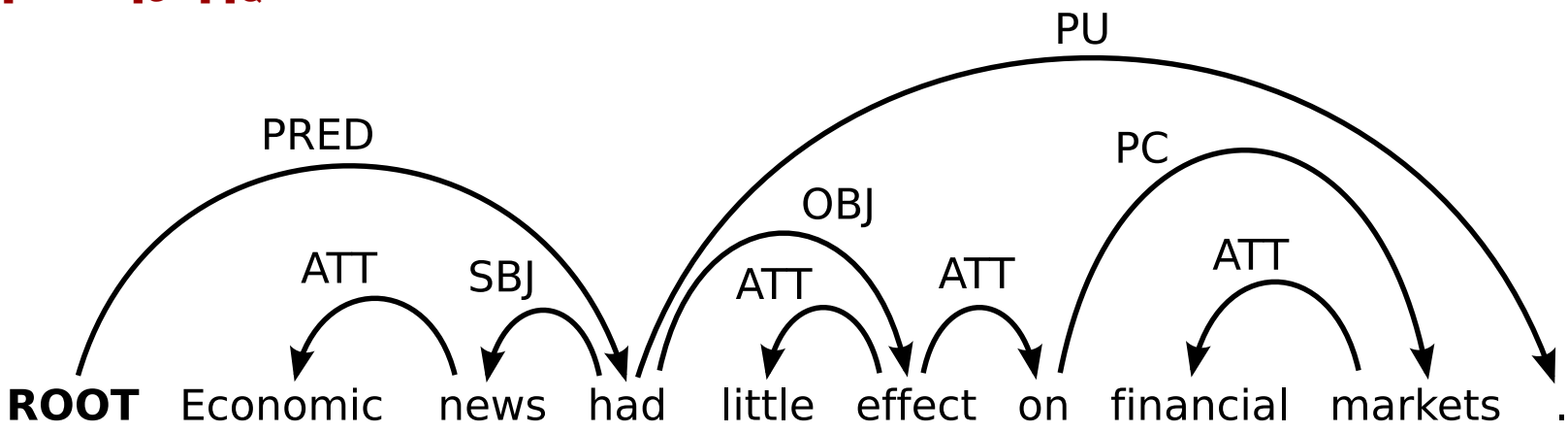
Example Transition Sequence

$[\textbf{ROOT}, \text{had}]_S \; [\,]_Q$



Example Transition Sequence

$[\text{ROOT}]_S \; [\,]_Q$



Deterministic Parsing

- Given an oracle o that correctly predicts the next transition $o(c)$, parsing is deterministic:

PARSE($w_1, \ldots, w_n$)
1 $c \leftarrow ([w_0]_S, [w_1, \ldots, w_n]_Q, \{\})$
2 **while** $Q_c \neq [\,]$ **or** $|S_c| > 1$
3 $t \leftarrow o(c)$
4 $c \leftarrow t(c)$
5 **return** $T = (\{w_0, w_1, \ldots, w_n\}, A_c)$

Oracles as Classifiers

- An oracle can be approximated by a (linear) classifier:

$$o(c) = \underset{t}{\operatorname{argmax}}\, \mathbf{w} \cdot \mathbf{f}(c, t)$$

- History-based feature representation $\mathbf{f}(c, t)$:
  - Features over input tokens relative to S and Q
  - Features over the (partial) dependency tree defined by A
  - Features over the (partial) transition sequence
- Weight vector $\mathbf{w}$ learned from treebank data:
  - Reconstruct oracle transition sequence for each sentence
  - Construct training data set $D = \{(c, t) \mid o(c) = t\}$
  - Maximize accuracy of local predictions $o(c) = t$

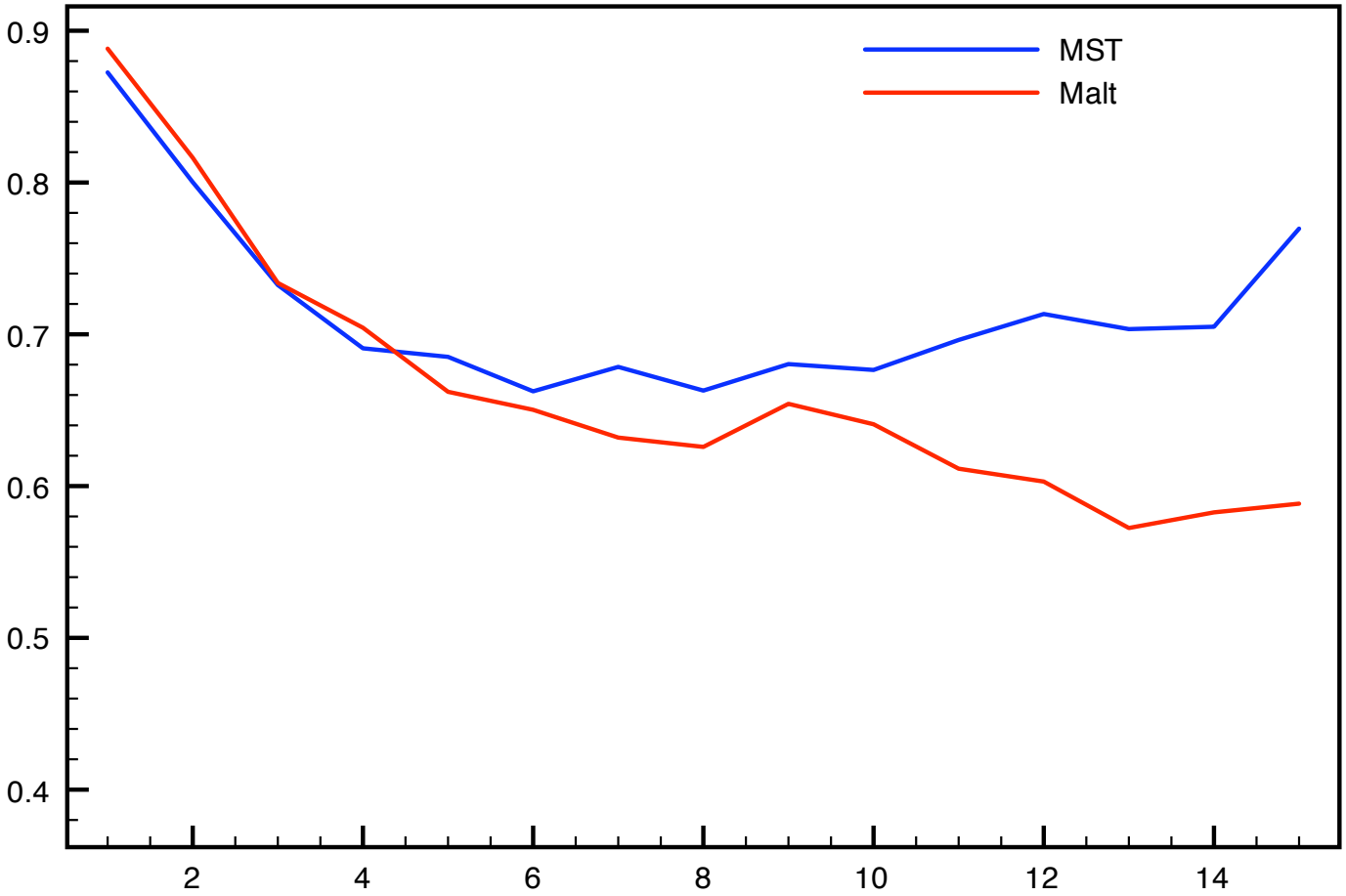
Deterministic Classifier-Based Parsing

- Advantages:
  - Highly efficient parsing – linear time complexity with constant time oracles and transitions
  - Rich history-based feature representations – no rigid constraints from inference algorithm
- Drawback:
  - Sensitive to search errors and error propagation due to deterministic parsing and local learning

Empirical Analysis

- CoNLL 2006 shared task [Buchholz and Marsi 2006]:
  - MaltParser [Nivre et al. 2006] – deterministic, local learning
  - MSTParser [McDonald et al. 2006] – exact, global learning
  - Same average parsing accuracy over 13 languages
- Comparative error analysis [McDonald and Nivre 2007]:
  - MaltParser more accurate on short dependencies and disambiguation of core grammatical functions
  - MSTParser more accurate on long dependencies and dependencies near the root of the tree
- Hypothesized explanation for MaltParser results:
  - Rich features counteracted by error propagation

Precision by Dependency Length



Beam Search and Structured Prediction

Beam Search

- Maintain the k best hypotheses [Johansson and Nugues 2006]:

PARSE($w_1, \ldots, w_n$)
1 BEAM $\leftarrow \{([w_0]_S, [w_1, \ldots, w_n]_Q, \{\})\}$
2 **while** $\exists c \in$ BEAM $[Q_c \neq [\,]$ **or** $|S_c| > 1]$
3 **foreach** $c \in$ BEAM
4 **foreach** t
5 ADD($t(c)$, NEWBEAM)
6 BEAM $\leftarrow$ TOP(k, NEWBEAM)
7 **return** $T = (\{w_0, w_1, \ldots, w_n\}, A_{\text{TOP}(1, \text{BEAM})})$

- Note:
  - Score$(c_0, \ldots, c_m) = \sum_{i=1}^{m} \mathbf{w} \cdot \mathbf{f}(c_{j-1}, t_j)$
  - Simple combination of locally normalized classifier scores
  - Marginal gains in accuracy

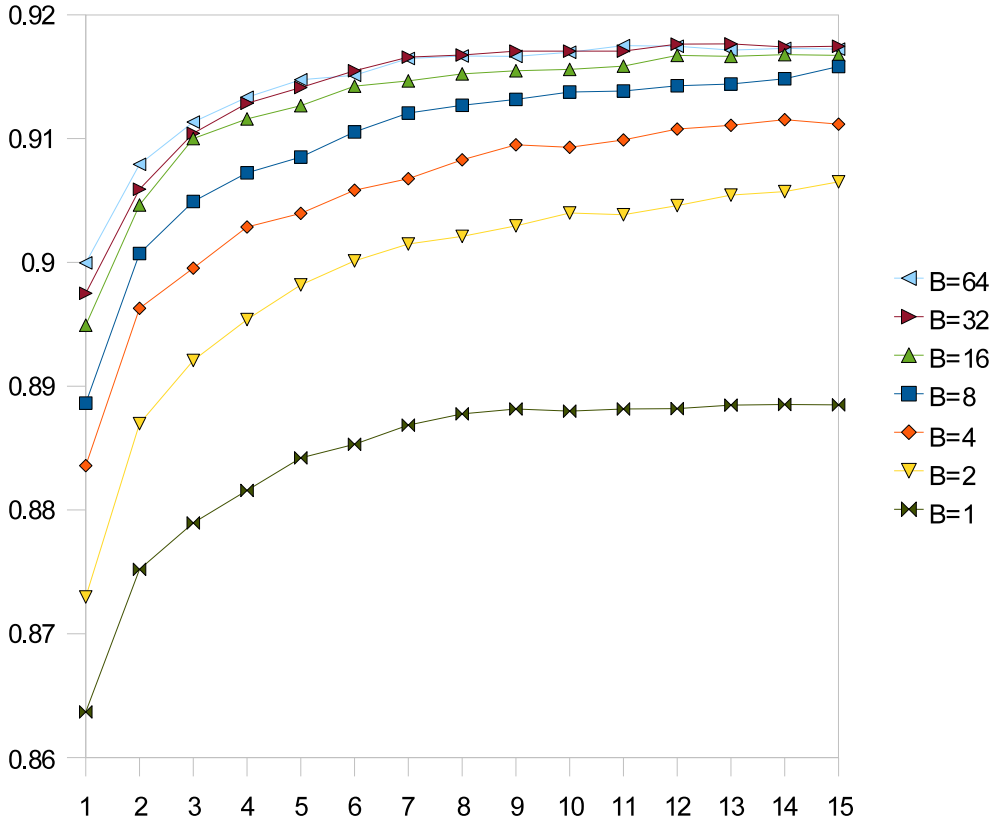
Structured Prediction

- Parsing as structured prediction [Zhang and Clark 2008]:
  - Minimize loss over entire transition sequence
  - Use beam search to find highest-scoring sequence
- Factored feature representations:

$$\mathbf{f}(c_0, \ldots, c_m) = \sum_{i=1}^{m} \mathbf{f}(c_{i-1}, t_i)$$

- Online learning from oracle transition sequences:
  - Structured perceptron [Collins 2002]
  - Early updates [Collins and Roark 2004]

Beam Size and Training Iterations

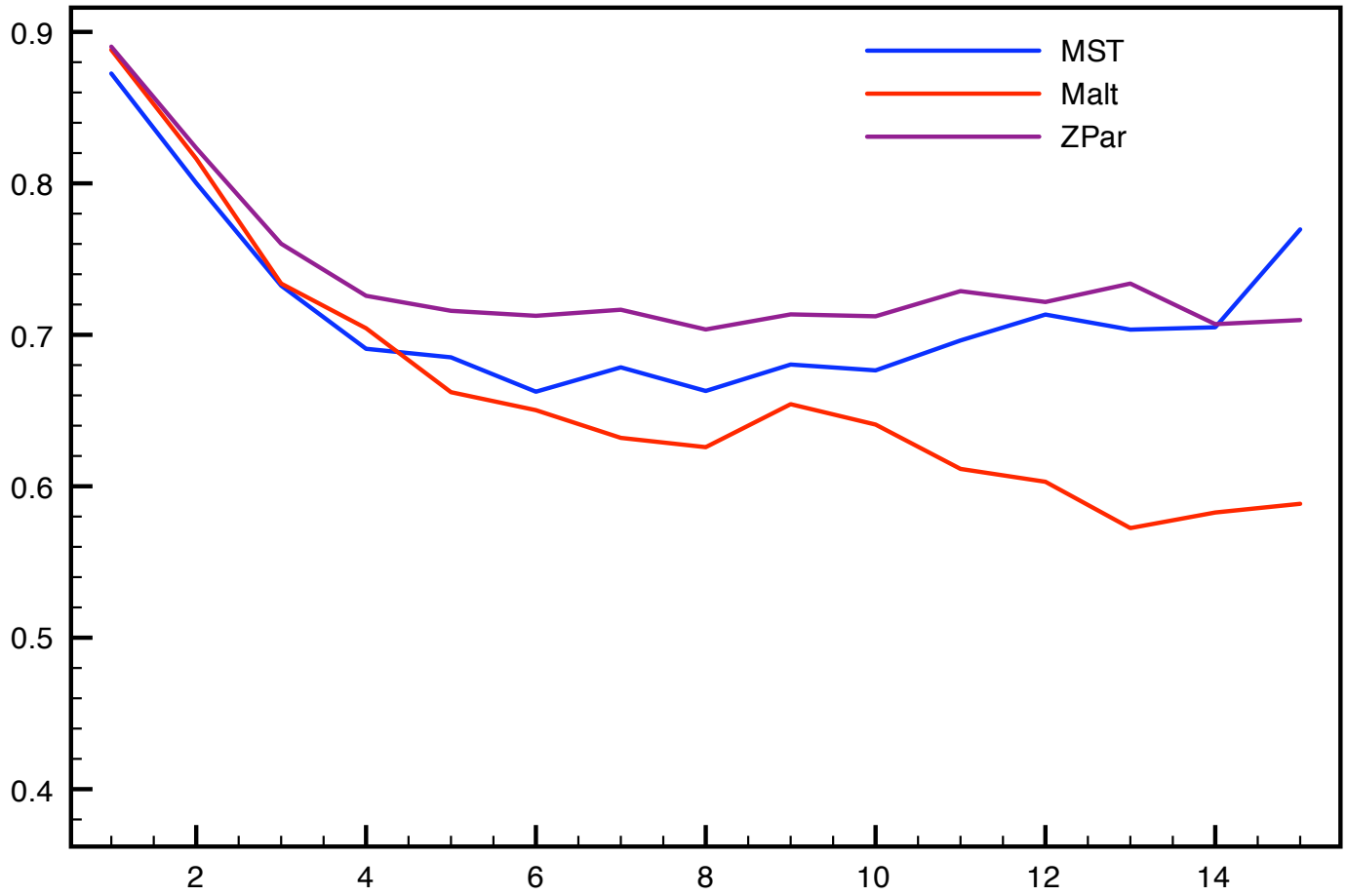


Yue Zhang and Stephen Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-Based and Transition-Based Dependency Parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 562–571.

The Best of Two Worlds?

- Like graph-based dependency parsing (MSTParser):
  - Global learning – minimize loss over entire sentence
  - Non-greedy search – accuracy increases with beam size
- Like (deterministic) transition-based parsing (MaltParser):
  - Highly efficient – complexity still linear for fixed beam size
  - Rich features – no constraints from parsing algorithm

Precision by Dependency Length Again



Even Richer Feature Models

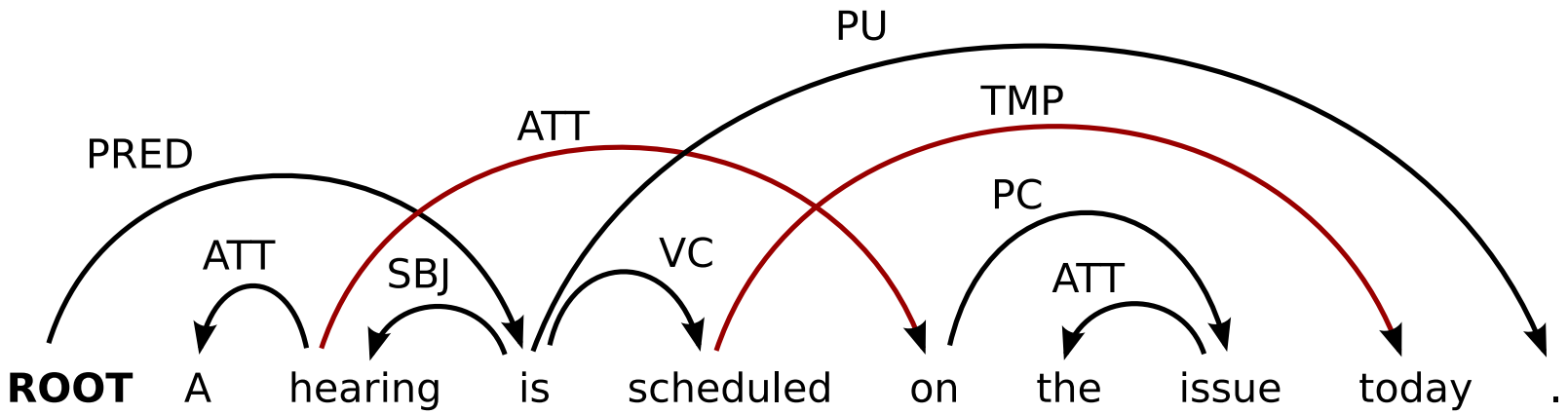
	ZPar	Malt
Baseline	92.18	89.37
+distance	+0.07	–0.14
+valency	+0.24	0.00
+unigrams	+0.40	–0.29
+third-order	+0.18	0.00
+label set	+0.07	+0.06
Extended	93.14	89.00

Yue Zhang and Joakim Nivre. 2011. Transition-Based Dependency Parsing with Rich Non-Local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 188–193.

Online Reordering for Non-Projectivity

Projectivity

- A dependency arc is projective if the head (transitively) dominates all intervening words
- Most dependency grammar theories do not assume projectivity (but many parsers do)



Non-Projectivity in Natural Language

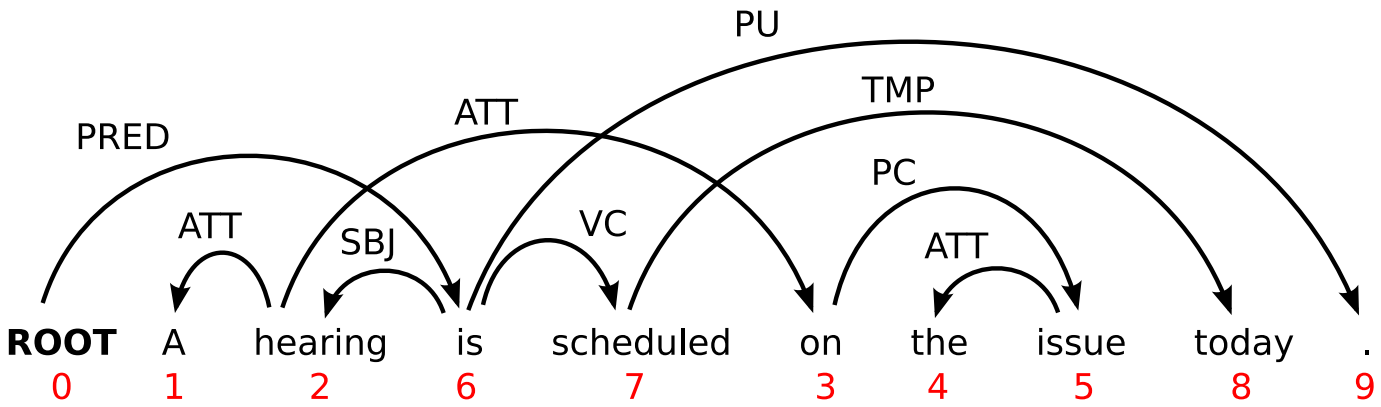
Language	Trees	Arcs
Arabic [Hajič et al. 2004]	11.2%	0.4%
Basque [Aduriz et al. 2003]	26.2%	2.9%
Czech [Hajič et al. 2001]	23.2%	1.9%
Danish [Kromann 2003]	15.6%	1.0%
Greek [Prokopidis et al. 2005]	20.3%	1.1%
Russian [Boguslavsky et al. 2000]	10.6%	0.9%
Slovene [Džeroski et al. 2006]	22.2%	1.9%
Turkish [Oflazer et al. 2003]	11.6%	1.5%

Transition-Based Approaches

- Pseudo-projective parsing [Nivre and Nilsson 2005]
  - Preprocess training data, post-process parser output
  - Approximate encoding with incomplete coverage
  - Relatively high precision but low recall
- Extended arc transitions [Attardi 2006]
  - Transitions that add arcs between non-adjacent subtrees
  - Increases nondeterminism in transition system
  - Infinitely many transitions needed for full coverage
- List-based algorithms [Nivre 2007]
  - Consider all word pairs instead of adjacent subtrees
  - Increases parsing complexity (and training time)

Projectivity and Word Order

- Projectivity is a property of a dependency tree only in relation to a particular word order
  - Words can always be reordered to make the tree projective
  - Given a dependency tree $T = (V, A, <)$, let the projective order $<_p$ be the order defined by an inorder traversal of T with respect to $<$ [Veselá et al. 2004]



Parsing with Online Reordering

- Add transition for reordering words [Nivre 2009]:
  - Swap $\frac{([\ldots, w_i, w_j]_S, \quad [\ldots]_Q, \quad A)}{([\ldots, w_j]_S, \quad [w_i, \ldots]_Q, \quad A)}$ $[0 < i < j]$
- Transition-based parsing with two interleaved processes:
  - Sort words into projective order $<_p$
  - Build dependency tree T by connecting adjacent subtrees
    - T is always projective with respect to $<_p$
    - T may be non-projective with respect to $<$

Example Transition Sequence

$[\textbf{ROOT}]_S$ [A, hearing, is, scheduled, on, the, issue, today, .]$_Q$

ROOT A hearing is scheduled on the issue today .

Example Transition Sequence

$[\textbf{ROOT}, \text{A}]_S$ [hearing, is, scheduled, on, the, issue, today, .]$_Q$

ROOT A hearing is scheduled on the issue today .

Example Transition Sequence

[**ROOT**, A, hearing]$_S$ [is, scheduled, on, the, issue, today, .]$_Q$

ROOT A hearing is scheduled on the issue today .

Example Transition Sequence

[**ROOT**, hearing]$_S$ [is, scheduled, on, the, issue, today, .]$_Q$

Example Transition Sequence

[**ROOT**, hearing, is]$_S$ [scheduled, on, the, issue, today, .]$_Q$

Example Transition Sequence

[**ROOT**, hearing, is, scheduled]$_S$ [on, the, issue, today, .]$_Q$

Example Transition Sequence

[**ROOT**, hearing, is, scheduled, on]$_S$ [the, issue, today, .]$_Q$

Example Transition Sequence

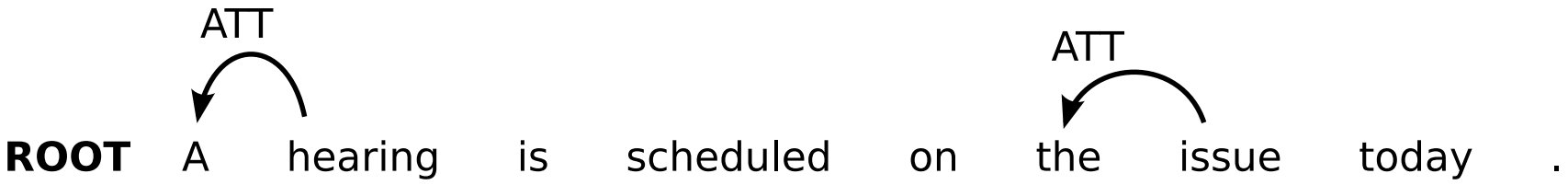
[**ROOT**, hearing, is, scheduled, on, the]$_S$ [issue, today, .]$_Q$

Example Transition Sequence

[**ROOT**, hearing, is, scheduled, on, the, issue]$_S$ [today, .]$_Q$

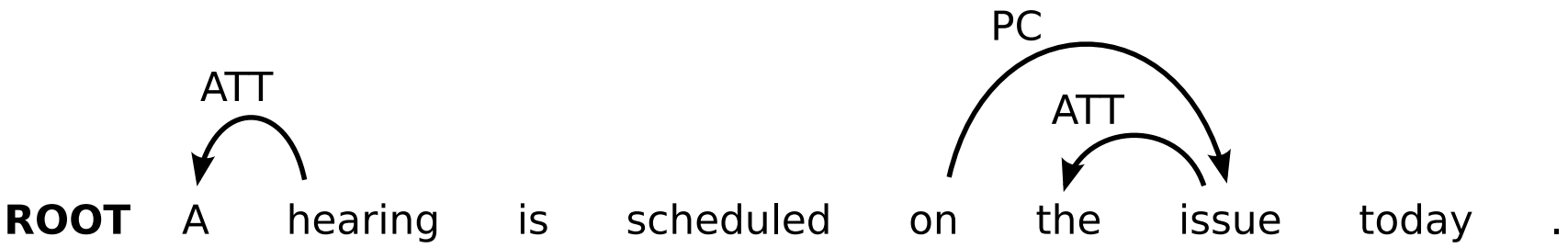
Example Transition Sequence

[**ROOT**, hearing, is, scheduled, on, issue]$_S$ [today, .]$_Q$



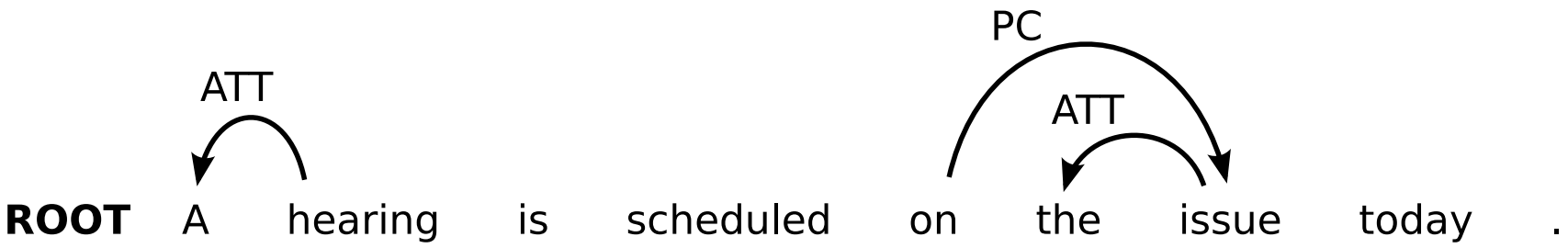
Example Transition Sequence

[**ROOT**, hearing, is, scheduled, on]$_S$ [today, .]$_Q$



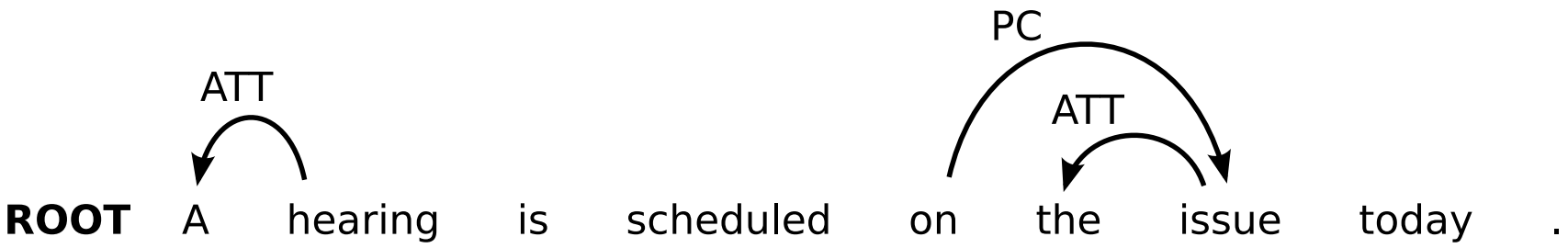
Example Transition Sequence

[**ROOT**, hearing, is, on]$_S$ [scheduled, today, .]$_Q$



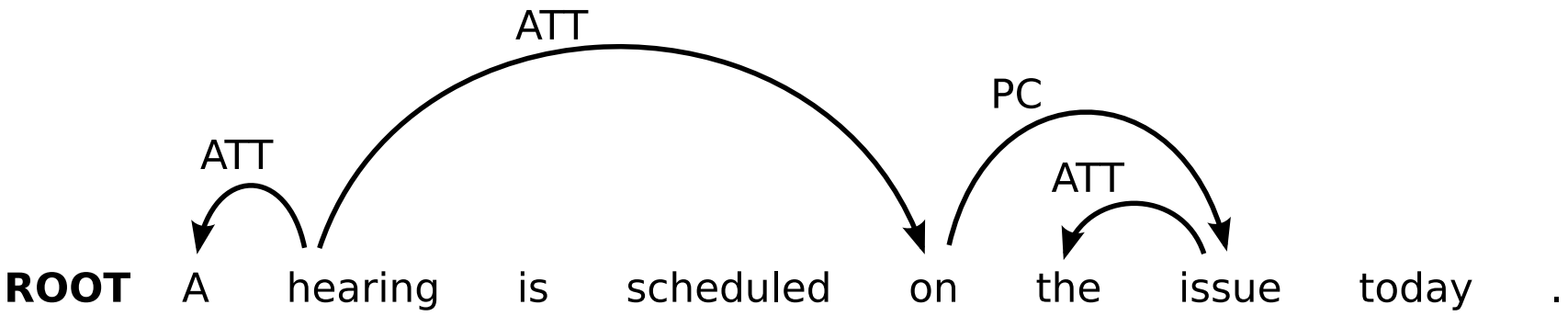
Example Transition Sequence

[**ROOT**, hearing, on]$_S$ [is, scheduled, today, .]$_Q$



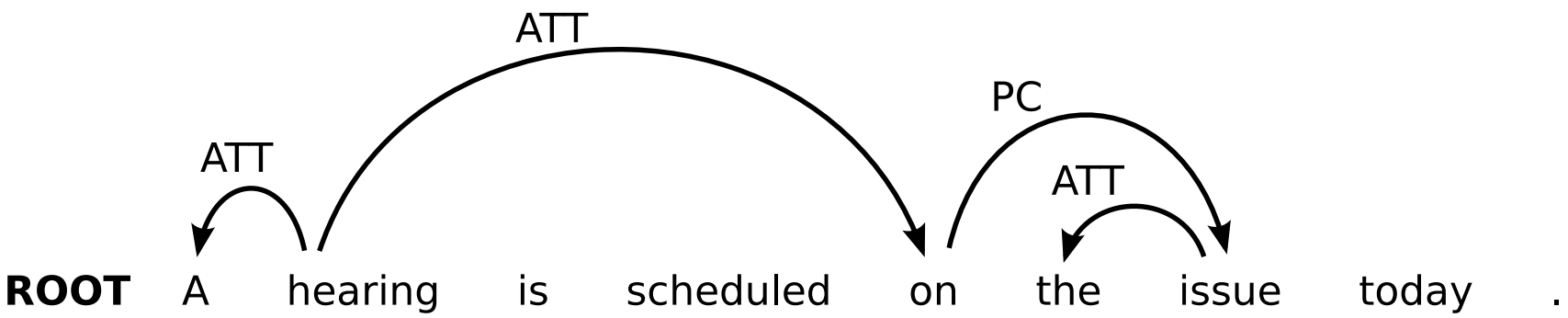
Example Transition Sequence

[**ROOT**, hearing]$_S$ [is, scheduled, today, .]$_Q$



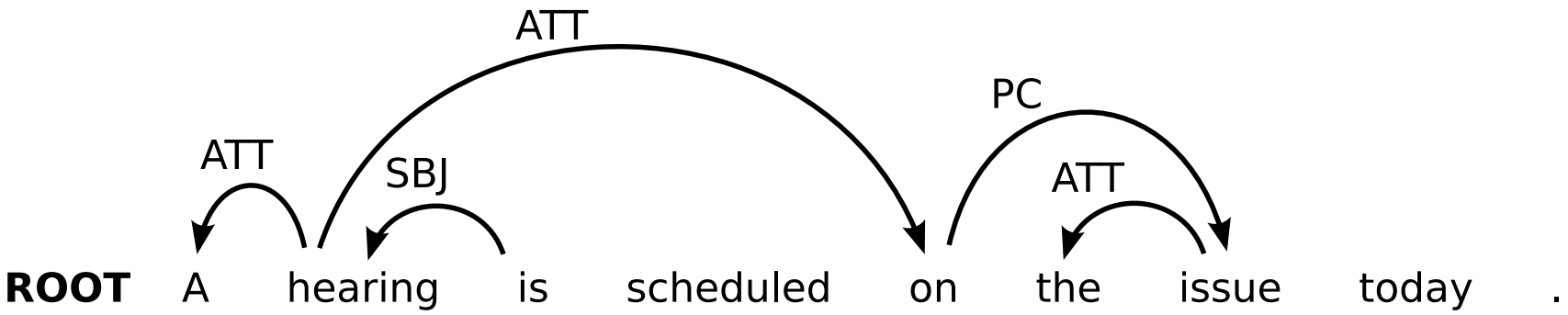
Example Transition Sequence

[**ROOT**, hearing, is]$_S$ [scheduled, today, .]$_Q$



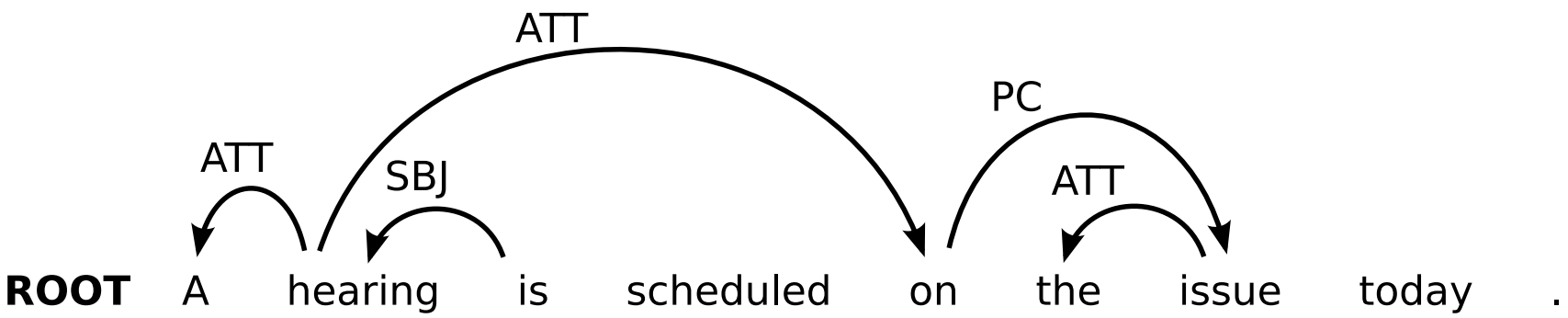
Example Transition Sequence

[**ROOT**, is]$_S$ [scheduled, today, .]$_Q$



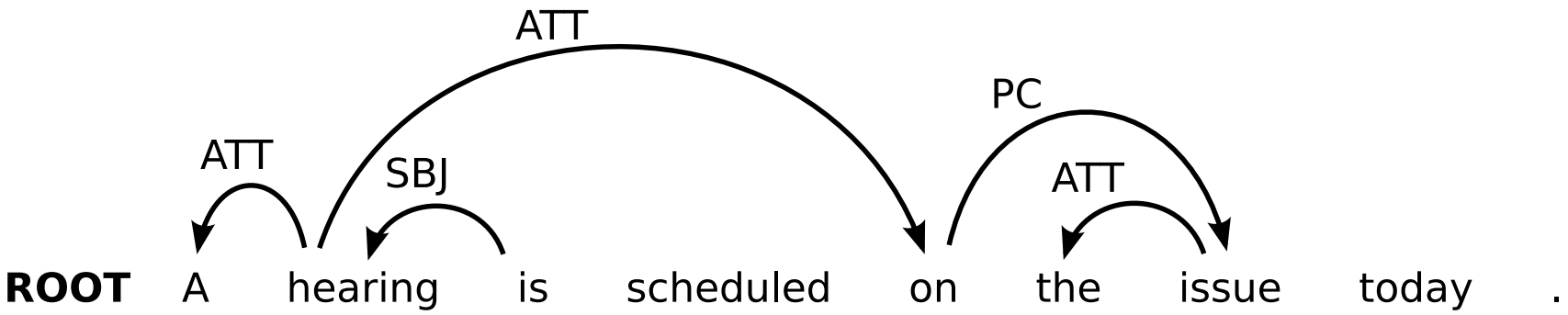
Example Transition Sequence

[**ROOT**, is, scheduled]$_S$ [today, .]$_Q$



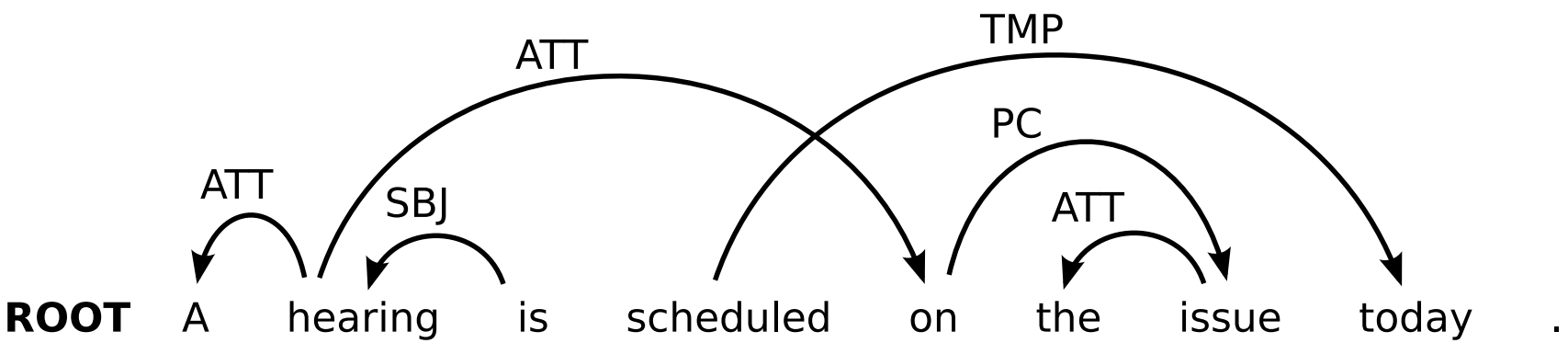
Example Transition Sequence

[**ROOT**, is, scheduled, today]$_S$ [.]$_Q$



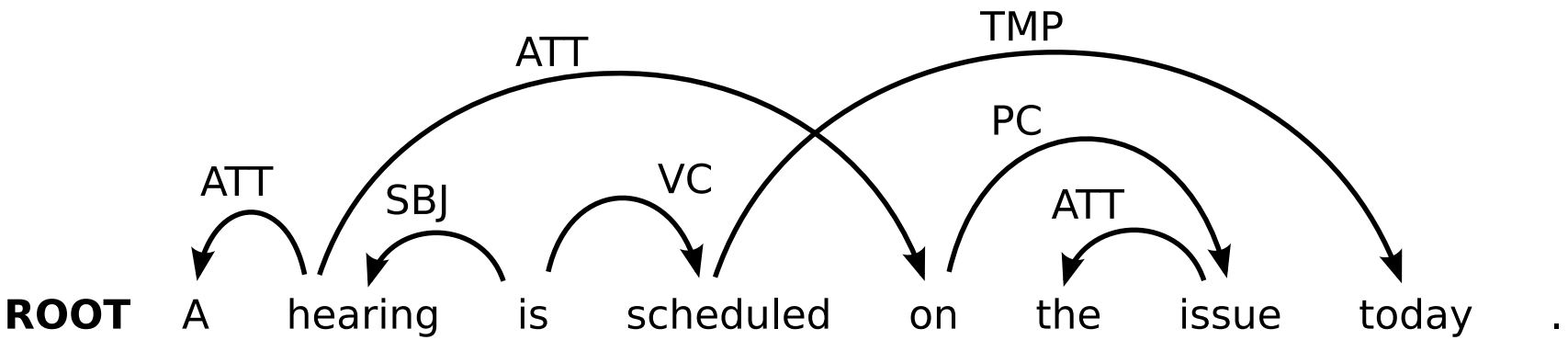
Example Transition Sequence

$[\textbf{ROOT}, \text{is}, \text{scheduled}]_S \; [.]_Q$



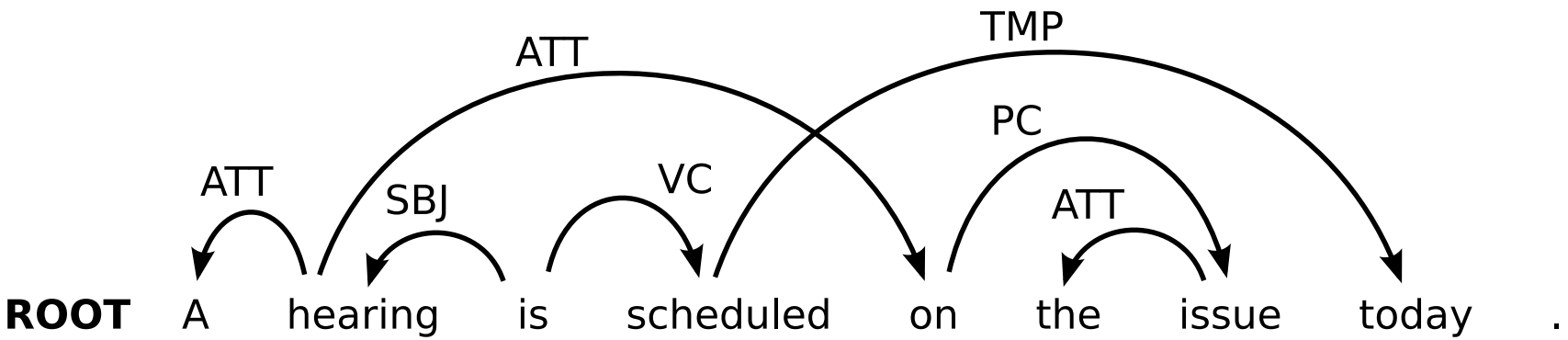
Example Transition Sequence

$[\textbf{ROOT}, \text{is}]_S \; [.]_Q$

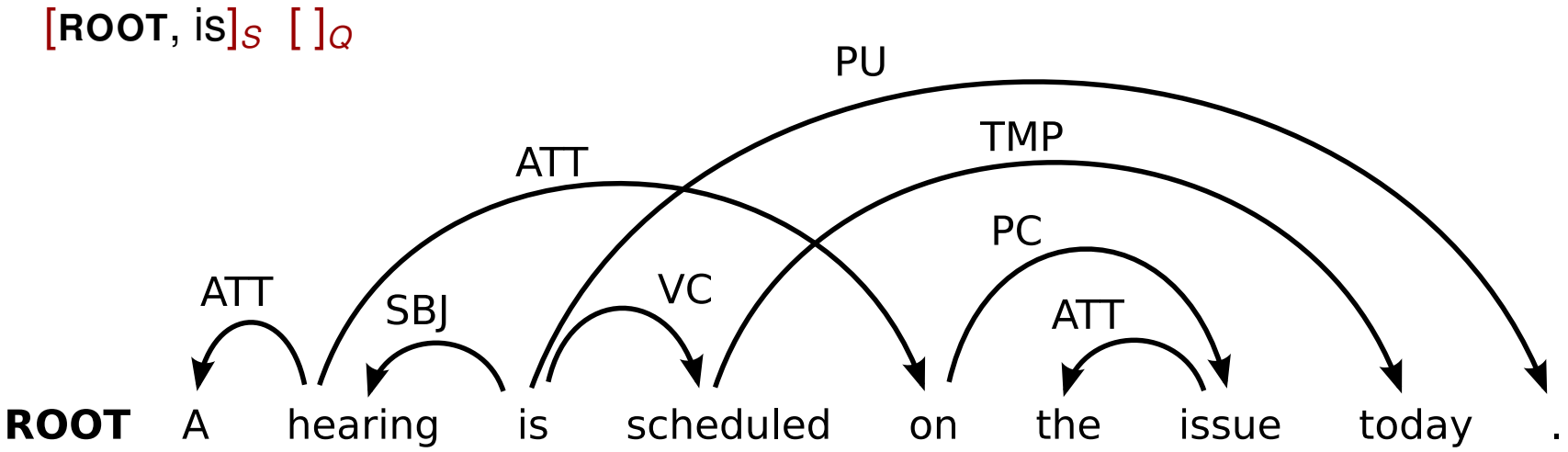


Example Transition Sequence

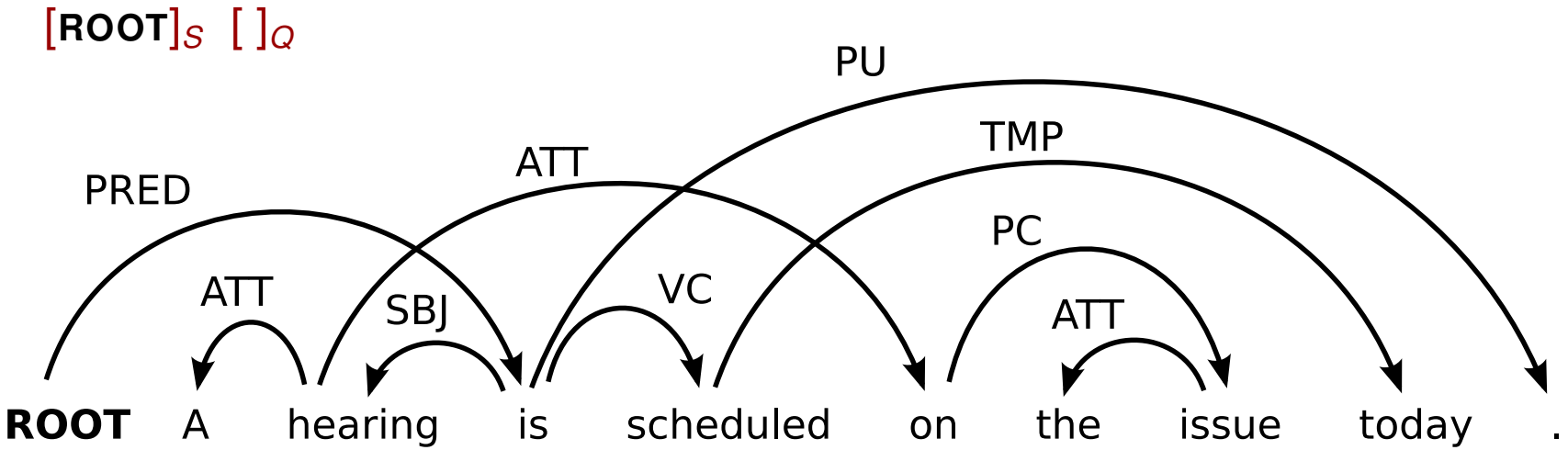
$[\textbf{ROOT}, \text{is}, .]_S \; [\,]_Q$



Example Transition Sequence



Example Transition Sequence



Empirical Results

- Deterministic transition-based parsing [Nivre 2009]:
  - Parsing in linear expected time (quadratic worst-case time)
  - Best results on Czech CoNLL 2006 data sets
- Beam search and structured prediction:
  - Joint work with Bernd Bohnet (unpublished)
  - Evaluation on CoNLL 2009 data sets (dev sets)

	Czech		German	
	LAS	UAS	LAS	UAS
Projective	80.8	86.3	86.2	88.5
Online reordering	83.9	89.1	88.7	90.9

Joint Tagging and Parsing

Part-of-Speech Tagging

- Part-of-speech tags in dependency parsing:
  - Crucially assumed as input, not predicted by the parser
  - Pipeline approach may lead to error propagation
  - Most PCFG-based parsers instead predict their own tags
- Recent interest in joint models for tagging and parsing:
  - Richly inflected languages [Lee et al. 2011]
  - Chinese [Li et al. 2011, Hatori et al. 2011]

Transition-Based Tagging and Parsing

- Redefine Shift transition [Hatori et al. 2011]:
  - Shift(p) $\dfrac{([\ldots]_S, \quad [w_i, \ldots]_Q, \quad A)}{([\ldots, w_i/p]_S, \quad [\ldots]_Q, \quad A)}$
- Transition-based parsing with three interleaved processes:
  - Tag words when they are shifted onto the stack
  - Optionally sort words into projective order $<_p$
  - Build dependency tree T by connecting adjacent subtrees

Empirical Results

- Joint tagging and parsing with online reordering:
  - Baseline = perceptron tagger + parser
  - Up to 3 tags per word allowed in joint model
  - Beam = 40 syntactic hypotheses + 4 tag variants
  - Evaluation on CoNLL 2009 data sets

	Chinese		Czech		English		German	
	LAS	TAcc	LAS	TAcc	LAS	TAcc	LAS	TAcc
Pipeline	77.0	92.8	82.5	99.1	89.3	97.6	88.1	97.2
Joint	77.3	93.3	82.6	99.3	89.7	97.8	88.3	97.8

Bernd Bohnet and Joakim Nivre. 2012. A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1455–1465.

Conclusion

Conclusion

- Transition-based parsing:
  - Efficient parsing using heuristic inference
  - Unconstrained history-based feature models
- Recent advances in synergy:
  - Beam search and structured prediction
  - Online reordering for non-projective trees
  - Joint part-of-speech tagging and parsing

References

- I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Díaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.
- 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.
- Igor Boguslavsky, Svetlana Grigorieva, Nikolai Grigoriev, Leonid Kreidlin, and Nadezhda Frid. 2000. Dependency treebank for Russian: Concept, tools, types of information. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 987–991.
- Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1455–1465.
- Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164.
- Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 112–119.
- Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–8.
- Sašo Džeroski, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdenek Žabokrtsky, and Andreja Žele. 2006. Towards a Slovene dependency treebank. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*.
- Jan Hajič, Barbora Vidova Hladka, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

References

- Jan Hajič, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. 2004. Prague Arabic Dependency Treebank: Development in data and tools. In *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*.
- Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint POS tagging and dependency parsing in chinese. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1216–1224.
- 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.
- Matthias Trautner Kromann. 2003. The Danish Dependency Treebank and the DTAG treebank tool. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 217–220.
- John Lee, Jason Naradowsky, and David A. Smith. 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 885–894.
- Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese POS tagging and dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1180–1191.
- Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.
- 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.
- Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

References

- 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. 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. 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.
- Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a Turkish treebank. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 261–277. Kluwer.
- P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proceedings of the 3rd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.
- Katerina Veselá, Havelka Jiri, and Eva Hajicová. 2004. Condition of projectivity in the underlying dependency structures. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 289–295.
- 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.